Sistemas Informáticos
Curso 2002-03

Entorno de simulación de la gestión de una FPGA bidimensional, parte de un sistema computador de propósito general basado en HW.

José María Moreno Juez
Fernando Valera Civantos
Juan Carlos Martín Corral

Dirigido por:
Julio Septién del Castillo
Departamento de Arquitectura de Computadores y Automática

Facultad de Informática
Universidad Complutense de Madrid

INDICE

Autorización

Por la presente, autorizamos a la Universidad Complutense de Madrid a difundir y utilizar con fines académicos, no comerciales, la memoria, el código y la documentación del presente proyecto de la asignatura de Sistemas Informáticos.

Juan Carlos Martín Corral Fernando Valera Civantos José María Moreno Juez

Resumen en español

FPGA Simulator 1.0 simula el comportamiento de una FPGA bidimensional dinámicamente reconfigurable. La simulación consiste en la gestión de una cola de tareas de tamaño rectangular (simple), con la posibilidad de multitarea, lo que permite ejecutar simultáneamente varias de estas tareas.

La ubicación de las tareas dentro de la FPGA se lleva a cabo mediante un algoritmo previamente implementado que gestiona el espacio libre en la FPGA. Este algoritmo gestiona las tareas en espera de ejecución, las que están ejecutándose actualmente en la FPGA y las que han terminado de ejecutar.

Nuestra solución FPGA Simulator 1.0 permite gestionar tanto la FPGA como la lista de tareas en espera a través de una sencilla e intuitiva interfaz gráfica. El algoritmo encargado de la ubicación de las tareas también puede ser configurado manualmente.
Entre la diversidad de algoritmos existentes (p.e. First-Fit (FF), Best-Fit (BF), First-Fit-Decreasing (FFD) y Best-Fit-Decreasing (BFD)), el algoritmo utilizado en nuestra solución es el Bin Packing MER, que ubica las tareas en el MER (Maximum Empty Rectangle) o Máximo Rectángulo Vacío.

Trabajar con distintas FPGAs y listas de tareas se convierte en una tarea sencilla gracias a la exportación de estos elementos a ficheros de texto, que permite tanto abrir y guardar como modificar los elementos involucrados en la simulación. A parte, el uso de ficheros incrementa la versatilidad del producto, permitiendo la exportación de elementos a otros ordenadores.

FPGA Simulator 1.0 permite además guardar la configuración de una simulación concreta de manera que se pueda reproducir en cualquier momento. Dado el carácter determinista de los algoritmos utilizados, una misma configuración desemboca en un mismo resultado, de forma que el almacenamiento de resultados se simplifica con el uso de un fichero de texto absolutamente independiente, que también es completamente portable.

A parte de la creación de un entorno de simulación, FPGA Simulator 1.0 proporciona además una idea general del comportamiento de la simulación por medio de métricas. Estas métricas, que deben ser utilizadas de manera orientativa, se basan en la utilización del espacio libre de la FPGA, y consisten en la ocupación media y la ocupación instantánea (en porcentaje) del espacio de la FPGA. La comparación de estas métricas como resultado de la ejecución de diferentes algoritmos sobre un mismo entorno de simulación nos puede dar un acercamiento al comportamiento real de los algoritmos.

La versión actual de FPGA Simulator 1.0 sólo está disponible para plataformas Windows (windows 95, 98, NT, 2000 y XP).

Implementado en Borland C++ Builder, este software presenta una gran estabilidad y rapidez de ejecución, al igual que un mínimo consumo de recursos de sistema. La posibilidad de incorporar manualmente nuevos algoritmos de ejecución convierten este producto en un software de gran utilidad en el campo de la investigación.

English summary

FPGA Simulator 1.0 simulates the behavior of a dynamically reconfigurable bidimensional FPGA. Simulation consists of a rectangular task queue management (simple tasks) with a multitask posibility, which implies the simultaneous execution of several tasks.

The task allocation within the FPGA is done through a previously programmed algorithm, which manages the free space in the FPGA. The task management involves the waiting for execution tasks, currently executing, and already finished ones

Our solution FPGA Simulator 1.0, enables the management of both the FPGA and the task queue through an easy and intuitive graphic interface. The algorithm used to allocate the tasks can also be manually configured. Between many other existing ones (i.e. First-Fit (FF), Best-Fit (BF), First-Fit-Decreasing (FFD) y Best-Fit-Decreasing (BFD)), Bin Packing MER was the final algorithm used in our product. Bin Packing MER allocates tasks in the Maximum Empty Rectangle available at a given moment.

The exportation of elements such as FPGAs and task queues to text files translates into a clear enhancement of the working capabilities, specially when working with different types of these kind of elements. The use of text files increases the product's versatility, as these elements can be exported to other computers with a minimum amount of space expense.

FPGA Simulator 1.0 allows saving concrete simulation configurations for later execution. Given that the algorithms used are deterministic, two identical configurations imply same results. Thus, results storage is simplified by the use of absolutely independent text files, which are also portable.

Apart from creating a simulation environment, FPGA Simulator 1.0 offers a general view of the algorithms behavior through the metrics. These metrics, which should be used only in an orientative way, are based on the available free space in the FPGA. The two metrics used are the current occupation and the average occupation (in percentage). The comparison of the metrics obtained out of the execution of different algorithms over the same simulation environment can give us an approximate idea of the real behavior of the algorithms.

The current version of FPGA Simulator 1.0 is only supported under Windows platforms (Windows 95, 98, NT, 2000 and XP).

Programmed under Borland C++ Builder, this software displays a great stability and execution time, at the same time as it consumes a minimum amount of system resources. The possibility of adding manually new algorithms provides this software with a great utility withing the scope of investigation.

Introducción al Hardware dinámicamente reconfigurable

Gestión de HWDR

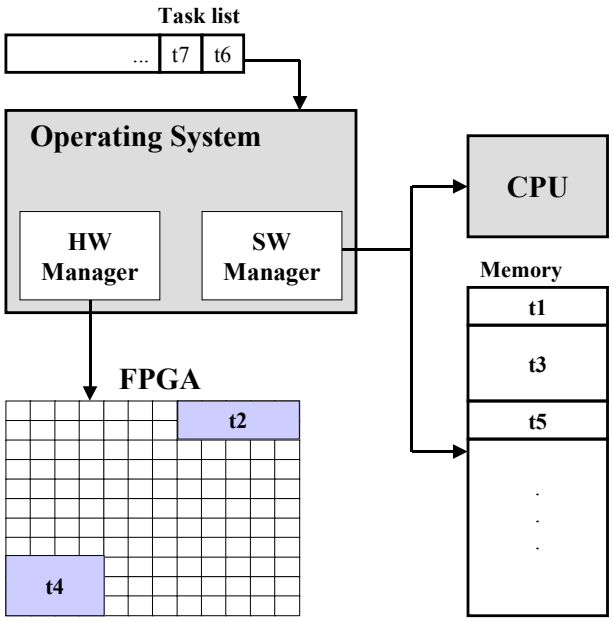
Gestión del HWDR: nueva funcionalidad del sistema operativo.

- Posibilidad de multitarea HW:
  - Los avances en tamaño permiten que quepan varias tareas diferentes simultáneamente en la FPGA: multiplexado del espacio, en vez de en el tiempo (p.e., Trimberger)

  - La reconfiguración dinámica y parcial permite que las distintas tareas HW entren y salgan de la FPGA independientemente unas de otras.

  - Recomendable una arquitectura de la FPGA homogénea y de grano fino.

- Aspectos de la gestión del HWDR por el SO:
  - Similitud con la gestión de la memoria para multitarea SW.
- Modelo genérico de sistema:

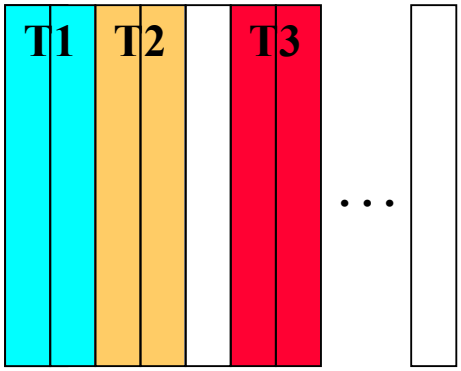


- Aspectos de la gestión de multitarea HW:
  - Organización de la FGPA (1D/2D)
  - Tamaño de las particiones (fijo/variable)
  - Requisitos de las tareas HW
  - Gestión de las tareas HW
  - Ubicación de las tareas
  - Carga de las tareas (aceleración)
  - Salida de las tareas
  - Defragmentación de la FPGA
  - Suspensión de tareas

- Organización de la FGPA (reconfiguración parcial)

  - Modelo en una dimensión (p.e., familia Virtex)

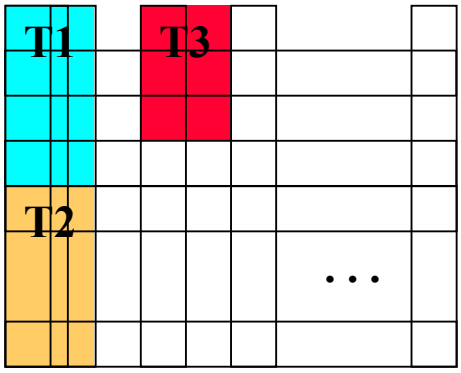
  La reconfiguración se hace por filas o columnas completas de CLBs ("frames").



  - Modelo de dos dimensiones (p.e., familia XC6200, o Virtex gestionada en 2D)

  La reconfiguración se hace por bloques de CLBs, a discreción del usuario.

  Puede hacerse en la Virtex, sin modificar los CLBs no afectados de cada columna.



- Tamaño de particiones fijo:
  - Similar a las páginas de memoria
  - Gestión muy simple
  - Problemas de espacio para tareas grandes
  - Fragmentación interna para tareas pequeñas
  - Posibilidad: incluir particiones de varios tamaños diferentes

- Tamaño de particiones variable:
  - Similar a los segmentos de memoria
  - Mayor flexibilidad
  - Permite ajustar su tamaño a los de las tareas exactamente
  - Mejor aprovechamiento del espacio libre
  - Fragmentación externa:

- Frecuente en 2D
- En 1D y 2D al salir tareas y dejar huecos
  - Necesidad de gestión dinámica y compleja de los recursos
  - Posibilidad: defragmentación

- Requisitos de las tareas HW:

  - Precompiladas: mapa de bits disponible
  - Reubicables: la estructura del interconexionado debe ser regular
  - Formas ("templates"):
    - Forma rectangular: más simple
    - Rotaciones.
    - Cambios de forma (aspecto).

    - Formas arbitrarias.
  - Múltiples formas para una misma tarea:

- Sólo distintos aspectos: mismos área y tiempo de ejecución
- Diversas áreas: tiempos de ejecución diferentes

- Gestión de las tareas HW:

- Características básicas de cada tarea (posibles):

  - T = (X, Y, t_ex, t_arr, t_max)
  - Si admite varias formas, cada una puede tener X, Y y un t_ex diferente.
    - t_arr = instante de llegada de la tarea
    - t_max = instante máximo permitido para que finalice la tarea
    - Aspectos de planificación
  - La tarea llega en t_ex
  - Si t_ex < (t_fin - t_arr) es posible demorar la ejecución de la tarea, en el intervalo desde t_init=t_arr hasta t_last=t_arr + (t_max - t_ex)
  - Razones de la demora: no hay sitio, o interesa dejar paso a otra tarea más urgente.
    Si se supera t_last ya no es posible ejecutar la tarea: descarte.

- Ubicación de las tareas HW:

- Posiciones fijas: asociadas a particiones fijas

- La información sobre el espacio libre es un único bit asociado a cada partición.
- Algoritmos de asignación simples: p.e., primera partición libre, o partición más aproximada en tamaño.
- Ubicación irrelevante dentro de la partición (p.e.: Bottom-Left, esquina inferior izquierda).

- Posiciones arbitrarias: asociadas a particiones variables.

- Mantenimiento de información sobre el espacio libre en estructuras complejas.
- Cualquier CLB puede ser candidato a ubicar la tarea.
- Algoritmos de asignación de una ubicación para la tarea: búsqueda del espacio libre.

- Carga de las tareas HW:

- Objetivo: minimizar el tiempo de carga de la configuración de cada tarea:

  - Prebúsqueda y precarga
    – El bitmap de la tarea planificada se busca y carga con antelación al instante de inicio

  - Compresión de configuraciones
    – Los bitmaps se cargan comprimidos
    – HW de descompresión en la FPGA

  - Caches de configuraciones
    – Pueden almacenarse por si vuelven a ejecutarse

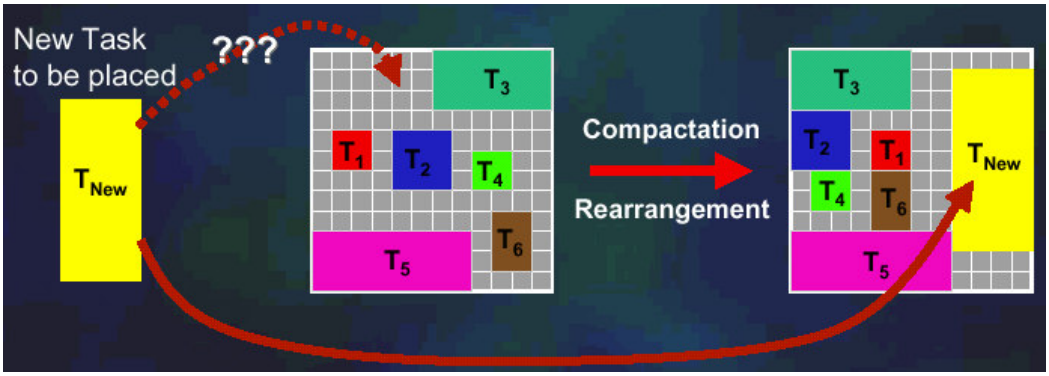
  - Múltiples contextos con reconfiguración parcial
    – Carga en un contexto mientras se ejecuta otro

  - Defragmentación del HW:

- Es necesaria cuando no hay ubicación posible para una nueva tarea, pero sucede que:

$$A(T_{New}) + \sum_{i=1}^{n} A(T_i) \leq A(FPGA)$$

- Compactación de las tareas en HW para hacer sitio a la nueva tarea:
- Desplazamiento de las tareas
- Recarga de las tareas


New Task
to be placed
???
T New
T 3
T 1
T 2
T 4
T 6
T 5
Compactation
Rearrangement
T 3
T 2
T 1
T New
T 4
T 6
T 5

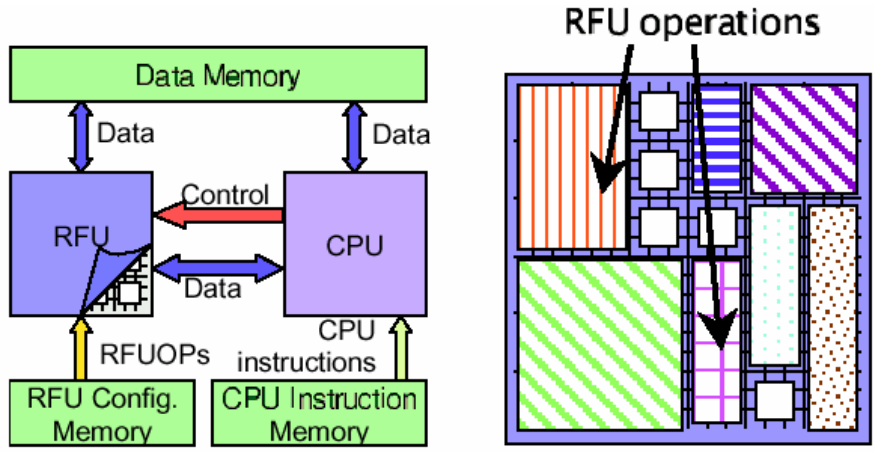
Trabajo en gestión de HWDR

- Principales áreas de investigación:
  - Asignación de ubicación (con/sin defragmentación):
    - Bazargan
    - Diessel
    - Compton
    - Walder

Bazargan

- Modelo de sistema:

-FGPA 2D, homogénea.
-Sistema mixto HW/SW



- Modelo de tareas HW (RFUOP):
  – Tareas rectangulares “a priori”: “hard templates”
  – Tareas organizadas jerárquicamente: “firm templates”
- Formadas por varias subtareas
- Las subtareas pueden reorganizarse para facilitar la ubicación de la tarea global

- En la práctica utilizan las de tipo “hard”

- Las tareas se ubican en posiciones arbitrarias, en particiones de tamaño variable.
- Uso de técnicas de "bin-packing": empaquetado de objetos (1, 2, o 3D) en contenedores de tamaño finito.

- Algoritmos de asignación:

  - Asignación "on-line" (bin-packing 2D): las tareas se deben procesar a medida que van llegando:
    - Interesa velocidad:
  - Algoritmos sencillos (FF, BF, etc.)
  - Estructuras de información de acceso rápido

    - Varias técnicas:
  - Información sobre rectángulos solapados
  - Información sobre rectángulos no solapados

  Asignación "off-line" (bin-packing 3D): se conoce de antemano el conjunto de tareas completo y se dispone de tiempo.

    - Algoritmos complejos de optimización:
  - Genéticos
  - Simulated annealing

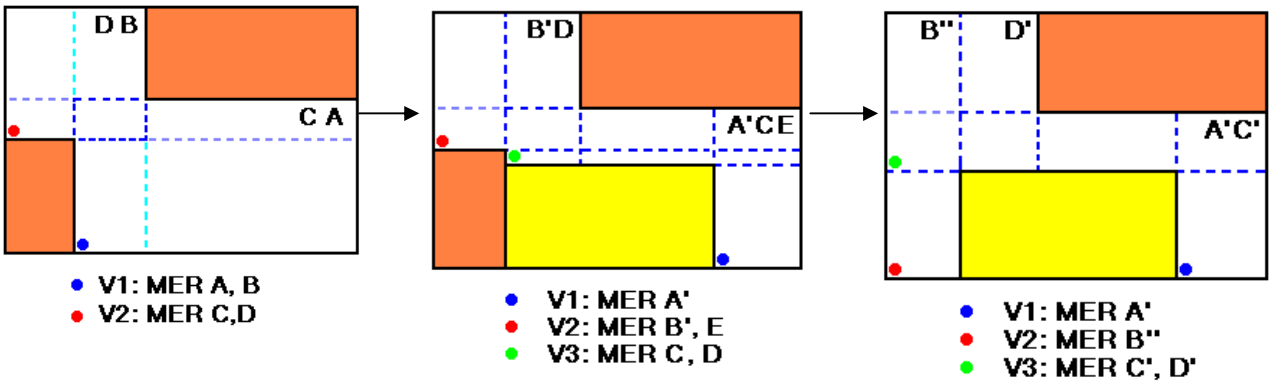
    - Asignación "on-line":
  - Algoritmo KAMER: "Keep All Maximum Empy Rectangles".

    - Se almacenan todos los rectángulos máximos vacíos (MER, maximum empty rectangles) en una lista.
    - Los MER pueden solaparse entre sí.
    - Cuando llega una tarea, se examina la lista de MER.
    - Si hay un hueco en la FGPA, está en algún MER.
    - Si no hay hueco, se rechaza la tarea (no hay defragmentación)

    - KAMER: complejidad elevada de mantenimiento de la lista de MER:
  - Elección del MER entre los posibles Amer > Atarea), mediante búsqueda:
      - First-Fit: el primero encontrado en el que cabe
      - Best-Fit: el que mejor se ajusta en tamaño (el más pequeño)
      - Worst-Fit: el que peor se ajusta en tamaño (el más grande)
      - Ubicación dentro del MER elegido: heurística Bottom-Left (esquina inferior izquierda).

- Actualización de la lista de MER cuando se generan nuevos MER:

- Al asignar un MER a una tarea, a partir del espacio sobrante se generan nuevos MER.
- Afecta a todos los otros MER con los que se solapa la tarea
- Al terminar una tarea hay que actualizar.



- La complejidad de mantenimiento de la lista es elevada (el nº de MER es de $O(N^2)$, con N = nº tareas).

- Asignación "on-line":

- Algoritmo basado en una lista de Rectángulos Vacíos No Solapados (NOER).
- Objetivo del nuevo algoritmo: mantener el número de rectángulos en O(N), con N = nº de tareas
- Resultado: 15 veces más rápido, 5% peor en ocupación de área.

- La complejidad de la mezcla e inserción es O(logN)
  - La complejidad total del algoritmo NOER es O (NlogN)

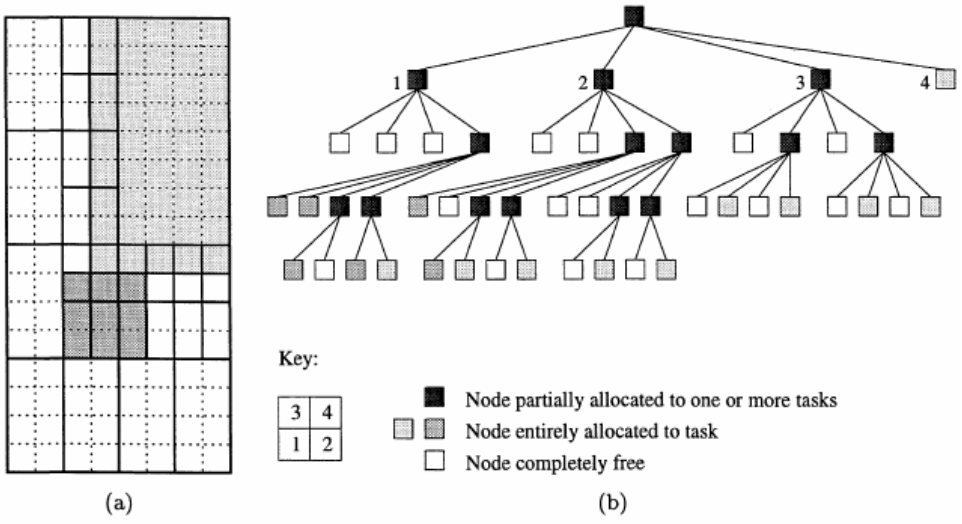
- Asignación "off-line":
  - Considera el tiempo como una tercera dimensión
  - Las tareas son cajas con volumen W*H*Tex.
  - Técnicas de bin-packing 3D

- Se intenta minimizar la altura (el tiempo total de ejecución) total del empaquetado.

- El tiempo de llegada se toma como una restricción a la ubicación de la caja de la tarea en el eje t.

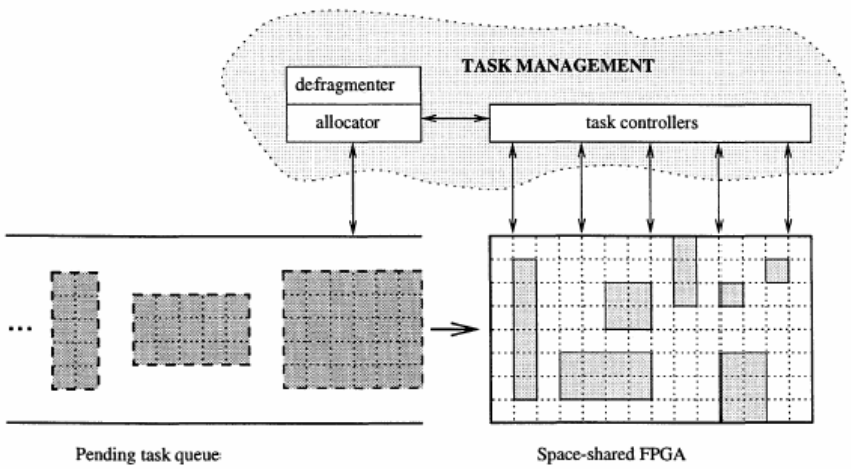
Diessel

- Modelo de sistema:
  - FGPA 2D, WxH bloques básicos, homogénea.
  - Particiones de tamaño variable y posiciones arbitrarias.
  - Tareas con forma rectangular, w*h.
  - Cola de tareas
  - Defragmentación.



- Asignación de ubicación para la tarea:

- Información sobre área libre organizada en Q-tree:



a) FPGA con dos tareas en ejecuFPGA con dos tareas en ejecuciónción
b) Q-tree con la información sobre área libre

– Se reduce la complejidad de la búsqueda.
– Puede haber área libre pero no utilizable: no todas las posiciones son candidatas.

- Defragmentación

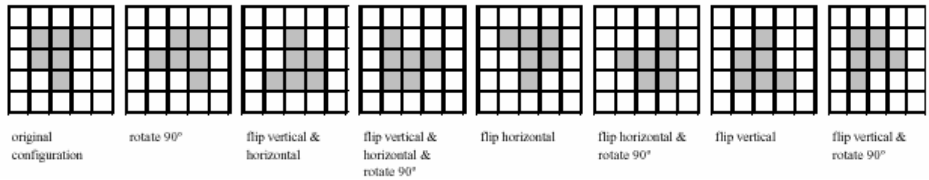
- Si no hay sitio disponible intenta defragmentar mediante reubicación de las tareas
- Técnica de “local repacking”: Reorganización del Q-tree.
- Busca un nodo con suficientes celdas libres como para que quepa la tarea
- Intenta reubicar las tareas que ocupan los nodos ocupados.
- Dentro del mismo nodo
- En otros nodos.

- Defragmentación

- Técnica de “ordered compation”
- Desplazamiento de las tareas en una dirección para hacer sitio
- Horizontal (a la derecha)
- Vertical (hacia arriba)

Compton

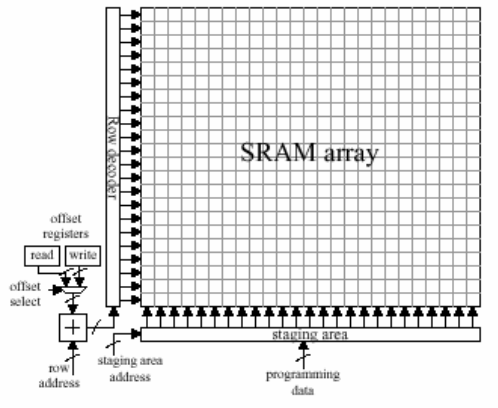
- Modelo inicial de FPGA: 2D, basada en Xilinx 6200
- Modelo de tarea:
  - Permutaciones posibles, combinables:



- Considera excesiva la complejidad de la gestión 2D.
- Nueva arquitectura R/D (Reubicación+Defragmentación), gestionada en una dimensión.

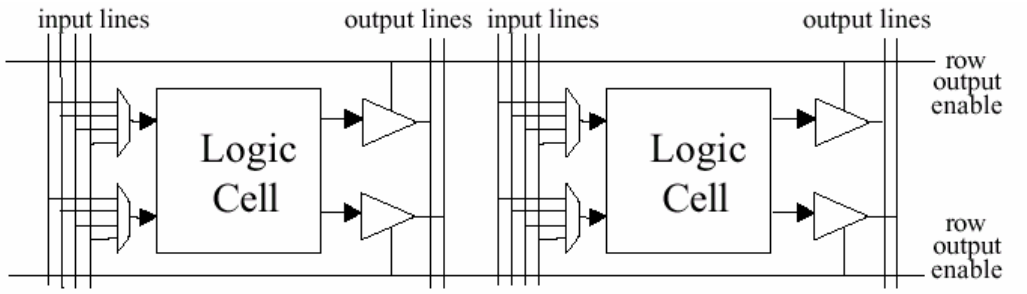
Características de la arquitectura R/D:

- Añade un buffer que permite mover filas completas.
- Cada fila se lee o escribe completa en una única operación a través del buffer.
- Una configuración: formada por una o varias filas.
- Dirección de fila: Número de fila dentro de una configuración
- Offset de fila (lectura y escritura): Desplazamiento de la primera fila en que se va a leer o escribir la configuración respecto de la primera fila de la FPGA.



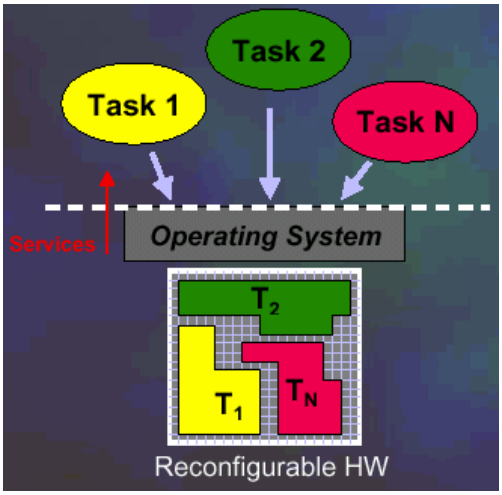
Entrada/salida virtualizada:

- E/S basada en buses globales, compartidos por todas las filas.
- Selección de la entrada por el multiplexor
- Selección de la salida por línea de habilitación para la fila (sólo una fila puede salir a una línea dada)
- Permite la reubicación de tareas
- Ejemplo: cuatro entradas y dos salidas:

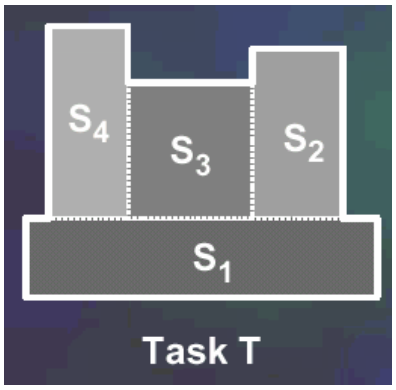


Walder

- Modelo de sistema:
  - El sistema operativo gestiona el HWDR 2D y la planificación, ubicación, carga y fin de las tareas HW:



- Modelo de tarea:
  - Formada por n subtareas rectangulares.
  - Subtareas contiguas (polyomino)
  - Transformaciones (a nivel de subtarea)



- Estrategias de ubicación:
  - First Fit (FF)
- Explora el array por filas y columnas hasta encontrar la primera ubicación libre, y coloca en ella la tarea.
  - Best Fit (BF)
- Calcula la fragmentación para cada posible ubicación de la tarea y escoge la mejor (menos fragmentada)

- Métrica de fragmentación: estimación en función del número de huecosde su área, normalizada:

$$F = 1 - \frac{\sqrt{\sum_i (n_i \cdot a_i^2)}}{\sum_i (n_i \cdot a_i)}$$

- Valores de la fragmentación para todos los CLBs candidatos (BF) al ubicar T3:

$$F = 1 - \frac{\sqrt{\sum_i (n_i \cdot a_i^2)}}{\sum_i (n_i \cdot a_i)}$$

Algoritmos Bin Packing

El Problema de Bin Packing

Descripción

Primeramente vamos a ver un ejemplo básico para entender rápidamente de que trata el problema de Bin Packing.

Dada una lista de *n* números reales $L = (w_1, w_2, ..., w_n)$, en que llamaremos $wi \in (0,1]$ al tamaño del ítem *i*, el problema consiste en asignar cada número a un canasto tal que la suma de los tamaños de los ítems dentro de un canasto no exceda **1**, y que a su vez se minimice el número de canastos a utilizar.

Discusión

El problema de Bin Packing se presenta en una variedad de problemas de embalaje y de producción. Por ejemplo, se desea saber cómo aprovechar lo mejor posible el espacio de carga en carros de tren o de camiones, para transportar cajas de manera de utilizar el mínimo número de carros.

Similar al caso de la mochila, es necesario revisar algunos casos particulares de fácil solución, los cuales quedan definidos por las siguientes preguntas:

1) ¿Cuál es la forma de los objetos?

La dificultad del problema depende fuertemente de la forma de los objetos a empacar. En efecto, empacar piezas de un puzzle es muy distinto a empacar cajas cuadradas en un contenedor. En problemas unidimensionales el tamaño de los objetos está dado simplemente por un número entero (lo cual lo convierte en un caso particular del Problema de la Mochila).

2) ¿ Hay restricciones acerca de la orientación y/o posición de los objetos?

Cuando se consolida carga en un contenedor, camión o carro ferroviario, muchas cajas tienen la señal *"This Side UP"*. Respetar esta restricción disminuye la flexibilidad de la solución y muy probablemente aumentará el número de unidades de transporte requeridas. Similarmente, cajas con material frágil suelen tener la etiqueta *"Do Not Stack"* y por lo tanto están restringidas a ubicarse en la cúspide de una pila

3) ¿ Se trata de un problema online u offline?

Es importante saber si se conoce la lista completa de objetos a empacar al comienzo del trabajo (problema offline), o bien la lista se va conociendo a medida que llegan los objetos para empacar y se va trabajando con la información que se tiene hasta el momento (problema online). Esta diferencia es muy importante porque se puede optimizar el proceso de empaque cuando se cuenta por anticipado con toda la información para poder planificar una estrategia. Así por ejemplo, podemos ordenar los objetos según su tamaño de mayor a menor para hacer más eficiente el uso de espacio.

Algoritmos de Solución

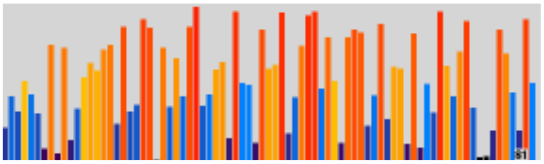
Desde la década de los 70's se han desarrollado varias heurísticas. Algunas de las más comunes son las llamadas *First-Fit (FF), Best-Fit (BF), First-Fit-Decreasing (FFD) y Best-Fit-Decreasing (BFD)*, todos ellos descritos en detalle en Johnson, D.S., A. Demers, J.D. Ullman, M.R. Garey y R.L. Graham*.

Las heurísticas FF y BF asignan ítems a canastos en el orden en que aparecen en la lista L sin utilizar información acerca de los ítems siguientes en la lista. Por ejemplo, FF funcionaría así: echar el ítem 1 en el canasto 1. Supongamos que estamos empacando el ítem j; ubicar el ítem j en el canasto de menor numeración cuyo contenido no exceda 1- wj.

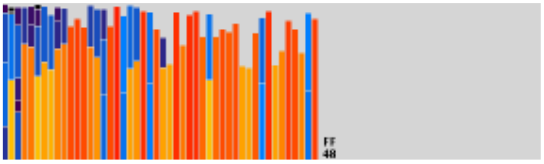
La heurística BF es similar a la FF pero echa el ítem j al canasto con el mayor contenido siempre que éste no exceda 1- wj. Este tipo de algoritmo es del tipo *online*. Las heurísticas FFD y BFD sin embargo, realizan un ordenamiento de la lista antes de operar. Así, FFD ordena los ítems de mayor a menor y luego realiza FF; la heurística BFD primero ordena la lista de mayor a menor y luego realiza BF. Este tipo de algoritmo es del tipo offline.

El Problema de Partición Entera puede considerarse como un problema de bin packing con dos canastos del mismo tamaño o un problema de mochila con una capacidad de la mitad del peso total de los objetos. Asi, los tres problemas están muy relacionados entre sí y son NP-completos.

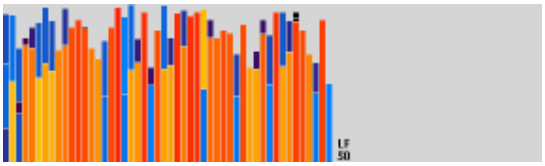
Todos los algoritmos que vamos a ver a continuación toman los objetos del estante de la cima en orden y los van poniendo uno a uno. Una nueva caja se crea siempre que el algoritmo no pueda usar una caja existente.



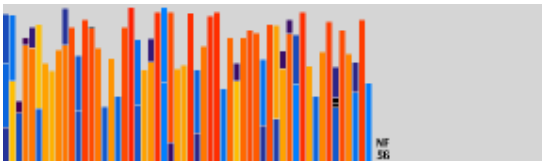
El conjunto inicial de objetos candidatos que se usan en las ilustraciones siguientes.



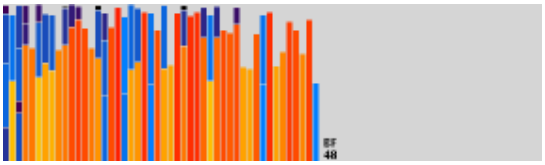
El algoritmo **First Fit** coloca un nuevo objeto en el canasto situado más a la izquierda que quepa.



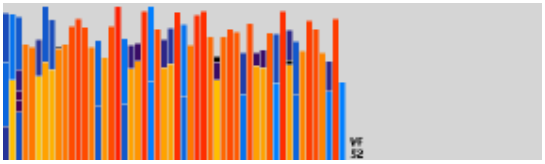
El algoritmo **Last Fit** coloca un nuevo objeto en el canasto situado más a la derecha que quepa.



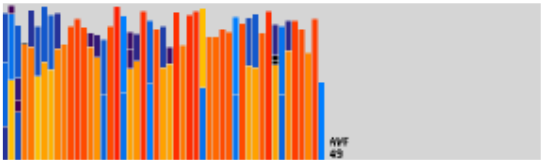
El algoritmo **Next Fit** coloca un nuevo objeto en el canasto situado más a la derecha, empezando un canasto nuevo si es necesario.



El algoritmo **Best Fit** coloca un nuevo objeto en el canasto más lleno.



El algoritmo **Worst Fit** coloca un nuevo objeto en el canasto más vacío.



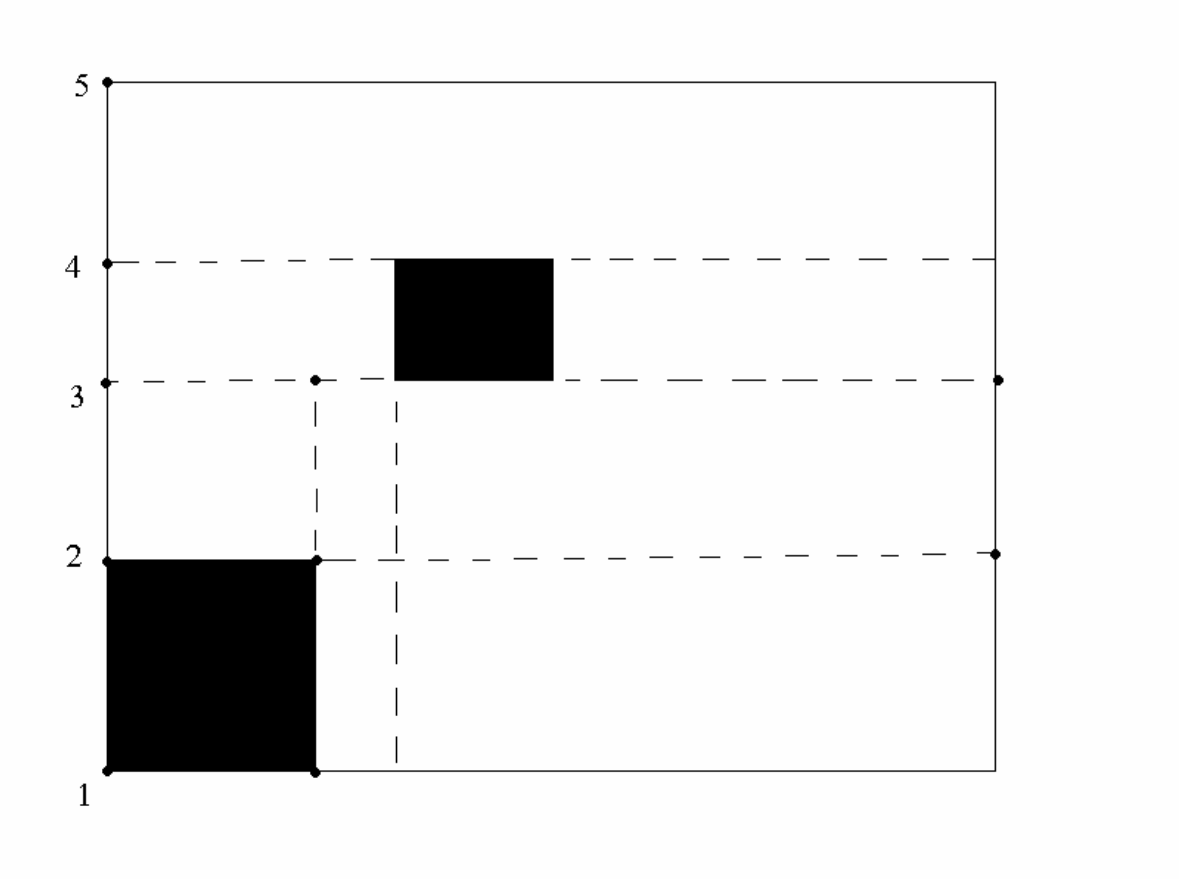
El algoritmo **Almost Worst Fit** coloca un nuevo objeto en el segundo canasto más vacío. (Este algoritmo es probablemente mejor que el Worst Fit y de mucho interés teórico).

Algoritmo MER

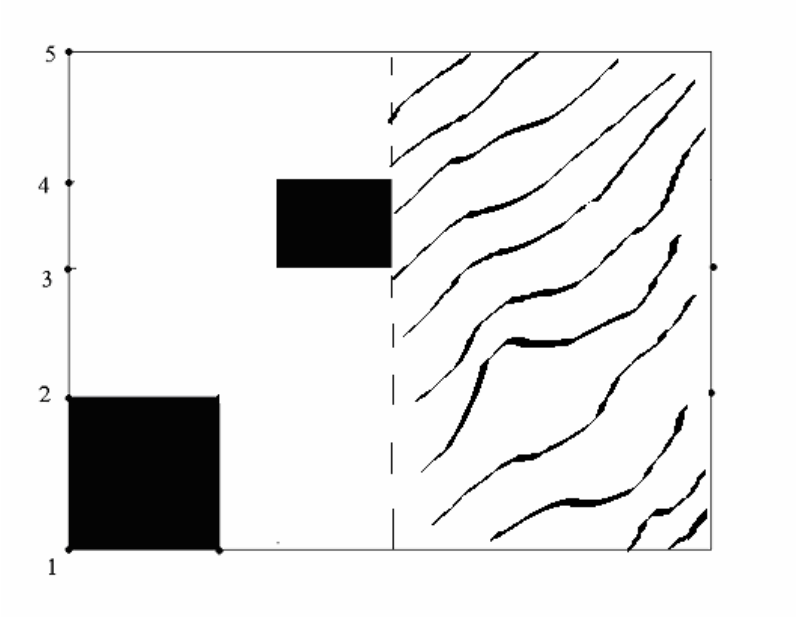
La explicación del algoritmo MER (maximum empty rectangle) se basa en la siguiente explicación simplificada. Cuando tenemos una FPGA cualquiera con una serie de tareas colocadas, lo primero que tenemos que hallar son los puntos de corte. Estos puntos de corte son los límites de la FPGA en la pared vertical

izquierda (1 y 5 en el dibujo) y los que resultan de llevar una recta paralela horizontal desde cada vértice de cada tarea hacia la pared izquierda (2, 3 y 4). De esta manera, vamos cogiendo todos las posibles combinaciones de pares de puntos y vamos obteniendo todos los rectángulos vacíos. Una vez que tenemos todos los rectángulos vacíos, nos quedamos con aquellos que son maximales, es decir, aquellos con mayor área. Estos rectángulos se van guardando en una lista que al principio se encuentra vacía. Esa tarea de ir eligiendo de ir seleccionando los rectángulos se lleva a cabo teniendo en cuenta 2 puntos principales:

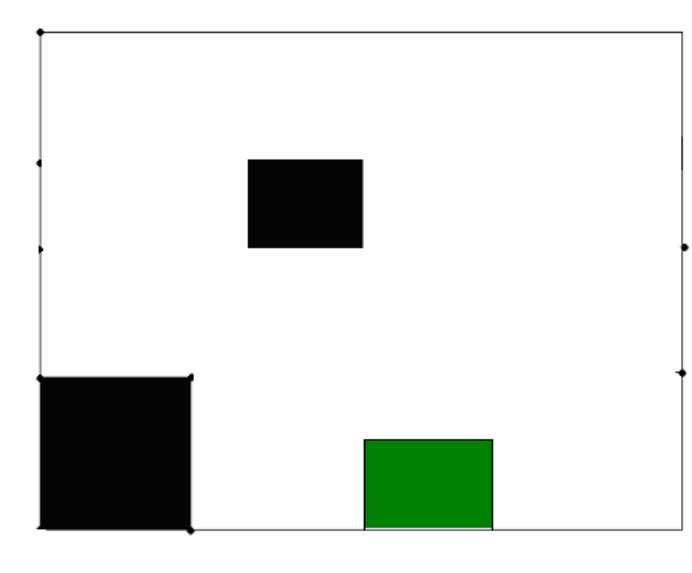
- La selección de rectángulos es de manera horizontal, es decir, se van trazando paralelas horizontales según los puntos de corte y vamos recolectando los rectángulos. En el ejemplo anterior llegamos a la conclusión de que el rectángulo máximo (es decir, el de mayor área) es el que aparece rayado en la siguiente figura.



- Nos quedamos con aquellos rectángulos maximales que NO se encuentren contenidos en la lista. Sería redundante introducir en la lista un rectángulo que es incluido por otro de la lista.



Así, en cada paso introducimos la tarea si cabe y ha llegado su tiempo de llegada en el máximo rectángulo vacío. De este modo si entra una nueva tarea en la FPGA (siempre que quepa) quedaría de la siguiente manera:



Ampliación de algoritmos

El proyecto está pensado para poder ampliar el numero de algoritmos con el que simular el problema de Binpacking. Para llevar a cabo esta ampliación, se deberán seguir unos sencillos pasos de programación. Para llevar a buen término estos pasos, será necesario poseer unos conocimientos básicos en programación orientada a objetos, y estar familiarizado con el entorno de borland para C++.
El primer paso será la implementación de la clase que representará nuestro algoritmo. La restricción que se impone es que esta clase será "hija" [1] de la interfaz *Algoritmo.h*[2]. Una vez realizada la clase, abriremos el proyecto con Borland C++ Builder. Hay que añadir la nueva clase recién creada al proyecto.

Para dar opción al usuario desde la interfaz gráfica de seleccionar el algoritmo a usar, deberemos añadir una entrada en el menú "Algoritmo -> Seleccionar". Ahora deberemos implementar el evento que se produce al hacer "clic" con el ratón. Podemos crear un nuevo formulario que muestre la lista de algoritmos existentes. Una vez realizada la selección del algoritmo, el último paso a realizar será la creación / inicialización de la variable global "**alg**", situada en la clase Ppal.h. En nuestro caso, la clase que implementa la clase Algoritmo.h se llama BinPackingMER, y la manera de crear e inicializar la variable "**alg**" es la siguiente:

```
this->alg = new BinPackingMER();
this->alg->Init(this->listaTareas,this->fpga->getH(),this->fpga->getW(),
      this->fpga->getName());
```

Por supuesto, es necesario recompilar el proyecto una vez realizados los cambios.

[1] Término en programación orientada a objetos que indica herencia entre clases, es decir, existe una clase padre de la que todas sus clases hijas heredan sus "metodos" ()funciones) y "atributos" (variables) públicos.
[2] Ver apéndice de código fuente.

Por último, si se quieren ampliar la funcionalidades de los algoritmos, deberemos retocar la interfaz de la clase “Algoritmo.h”, para añadir nuevos métodos.

Ejecución del algoritmo

Vamos a ver un ejemplo de la ejecución del algoritmo.

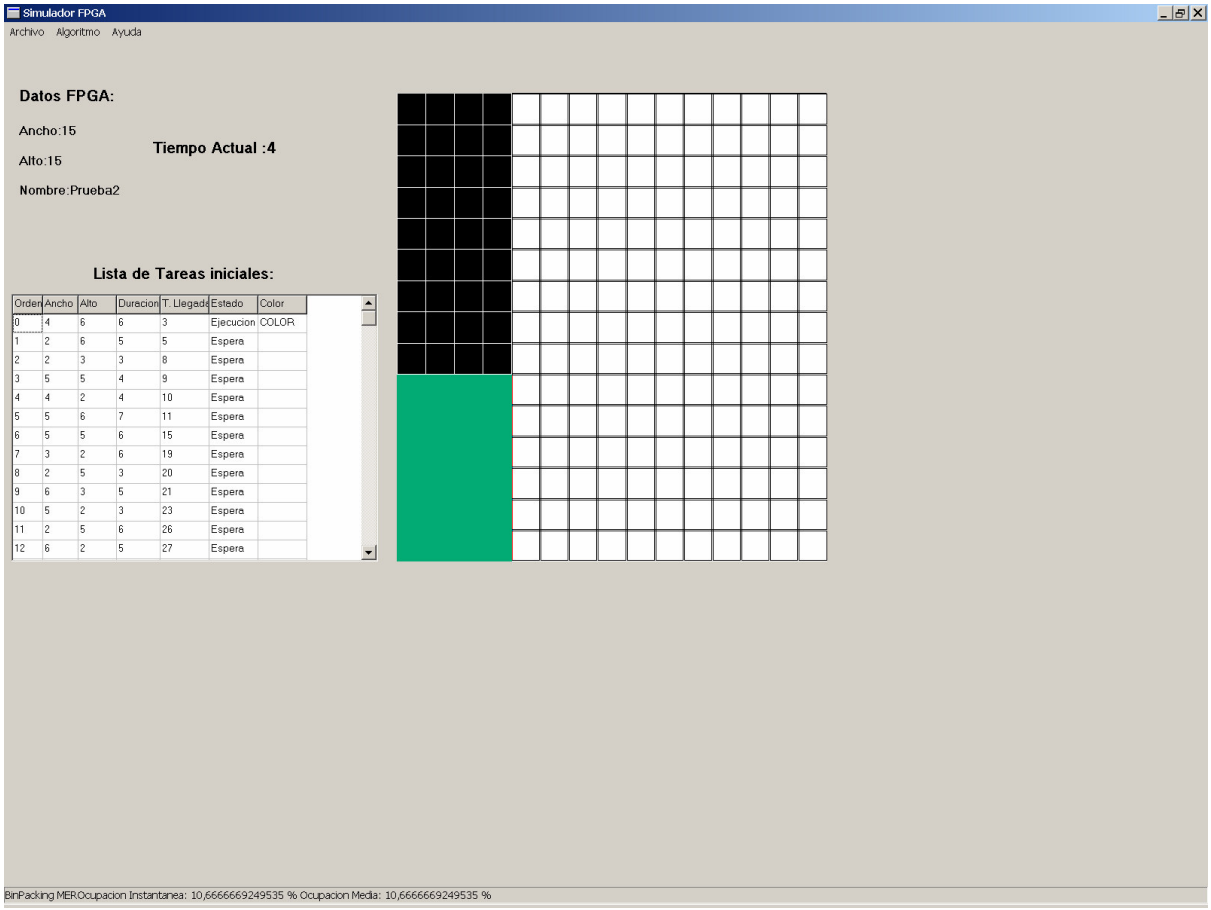
FPGA *Ancho* 15

Alto 15

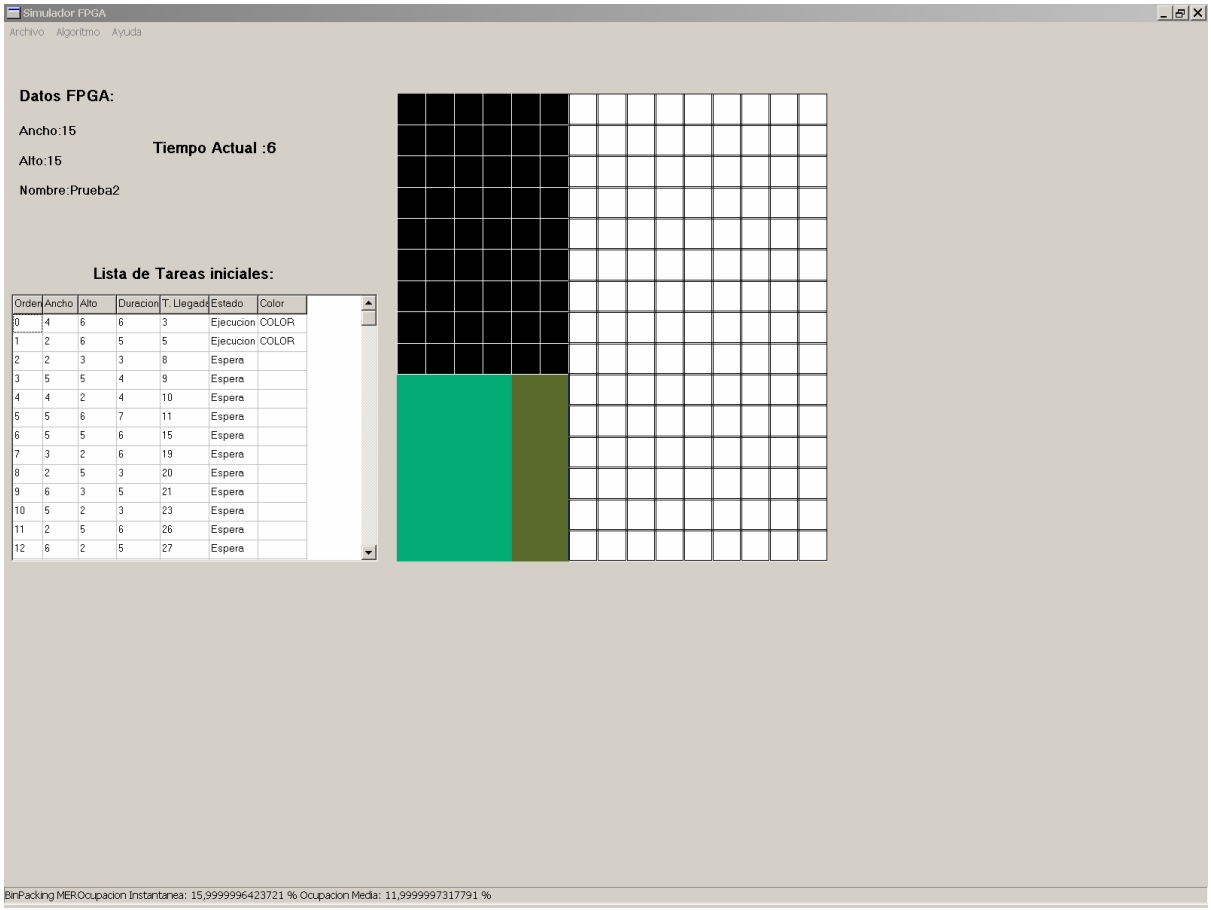
TAREAS

Alto	***Ancho***	***Tiempo llegada***	***Duración***
4	6	3	6
2	6	5	5
2	3	8	3
5	5	9	4
4	2	10	4
5	6	11	7
5	5	15	6
3	2	19	6
2	5	20	3
6	3	21	5
5	2	23	3
2	5	26	6
6	2	27	5
6	3	30	2
2	3	31	5
6	6	34	6
3	7	37	4
2	3	39	4
3	6	41	8
3	7	46	5
5	5	47	5

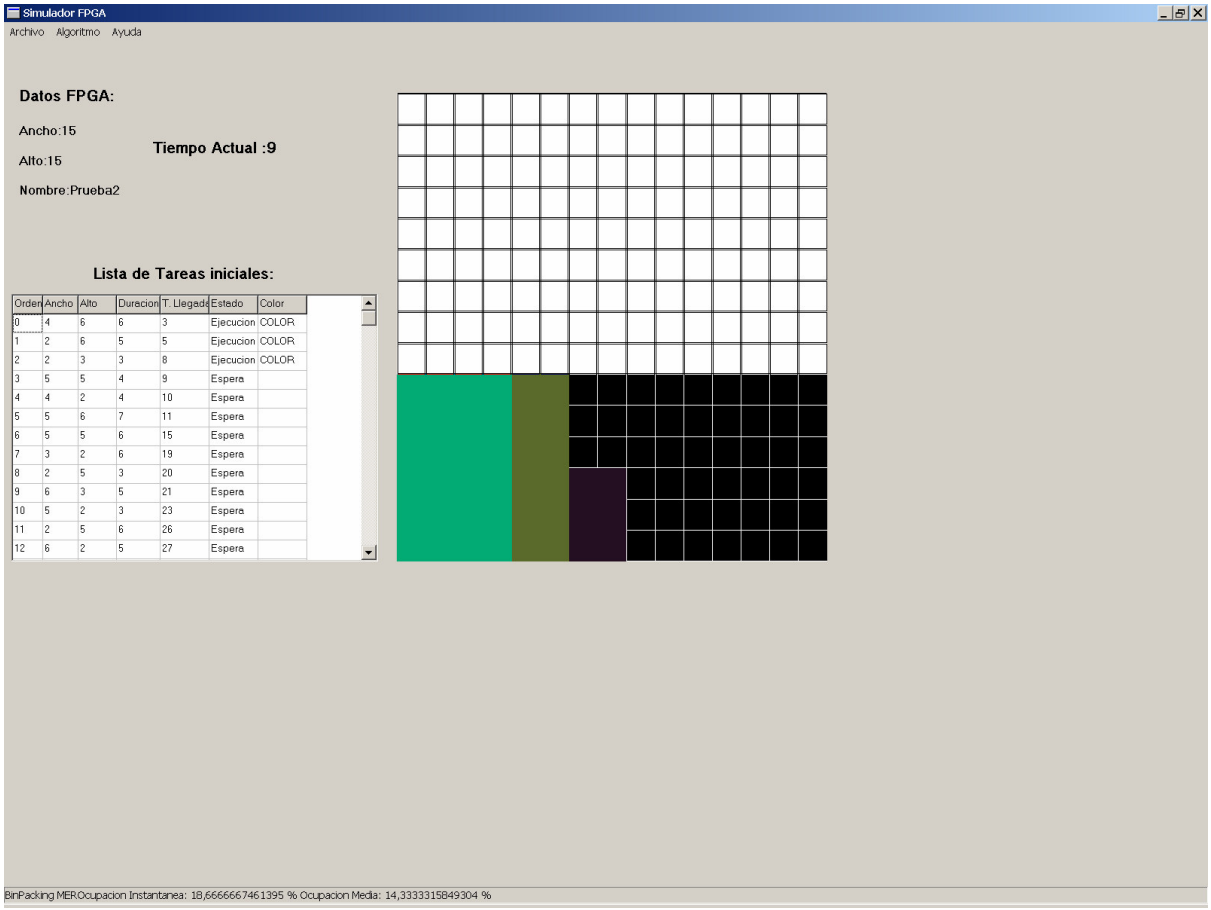
Entra la primera tarea en la FPGA en el instante t=4. El rectángulo sombreado en blanco es el máximo donde debe situarse la siguiente tarea en entrar.



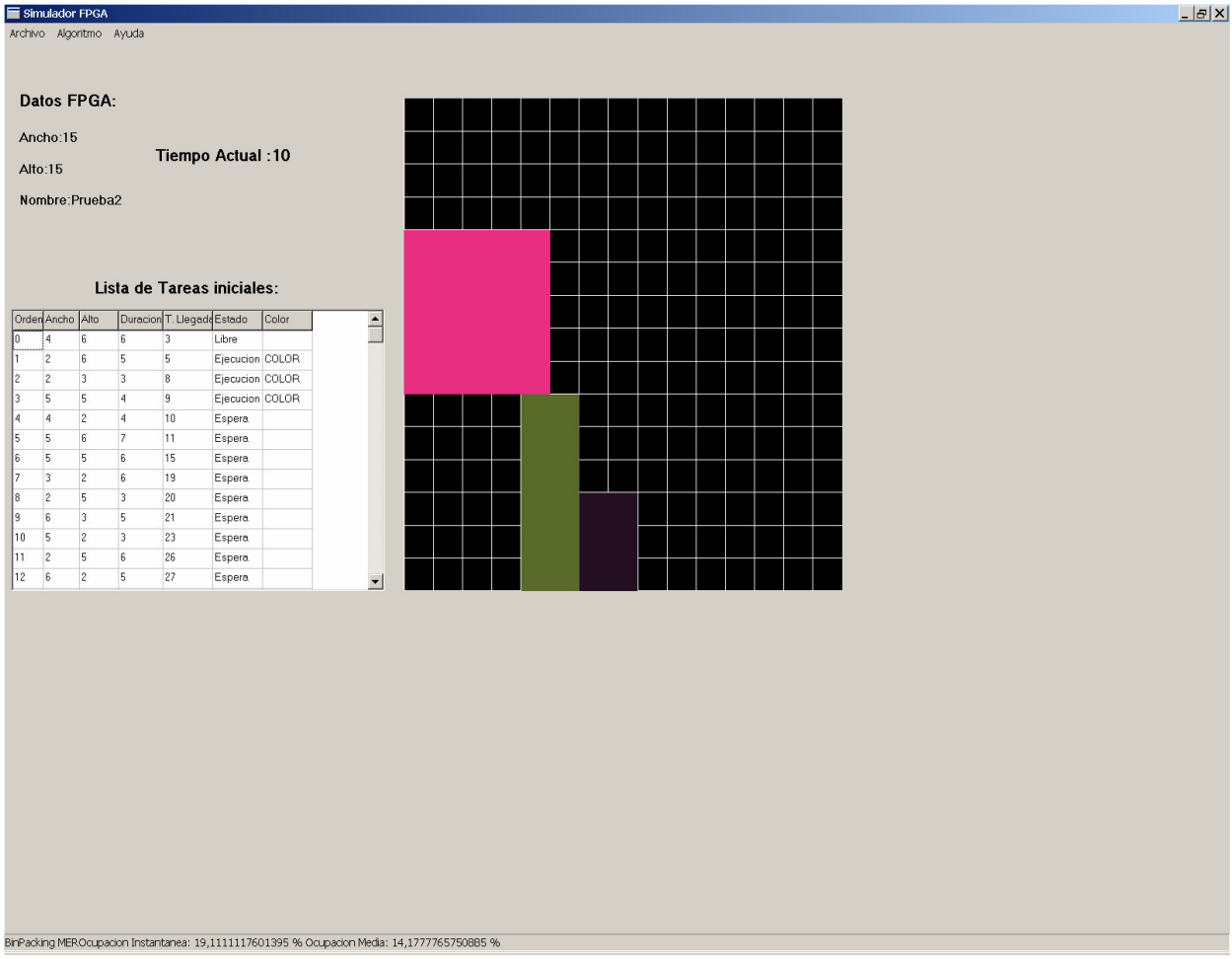
Ahora entra la segunda tarea (Orden 1) y se coloca en el rectángulo máximo anteriormente indicado. Simplemente señalar que la tarea siempre se sitúa en los vértices inferiores izquierdos del rectángulo máximo.



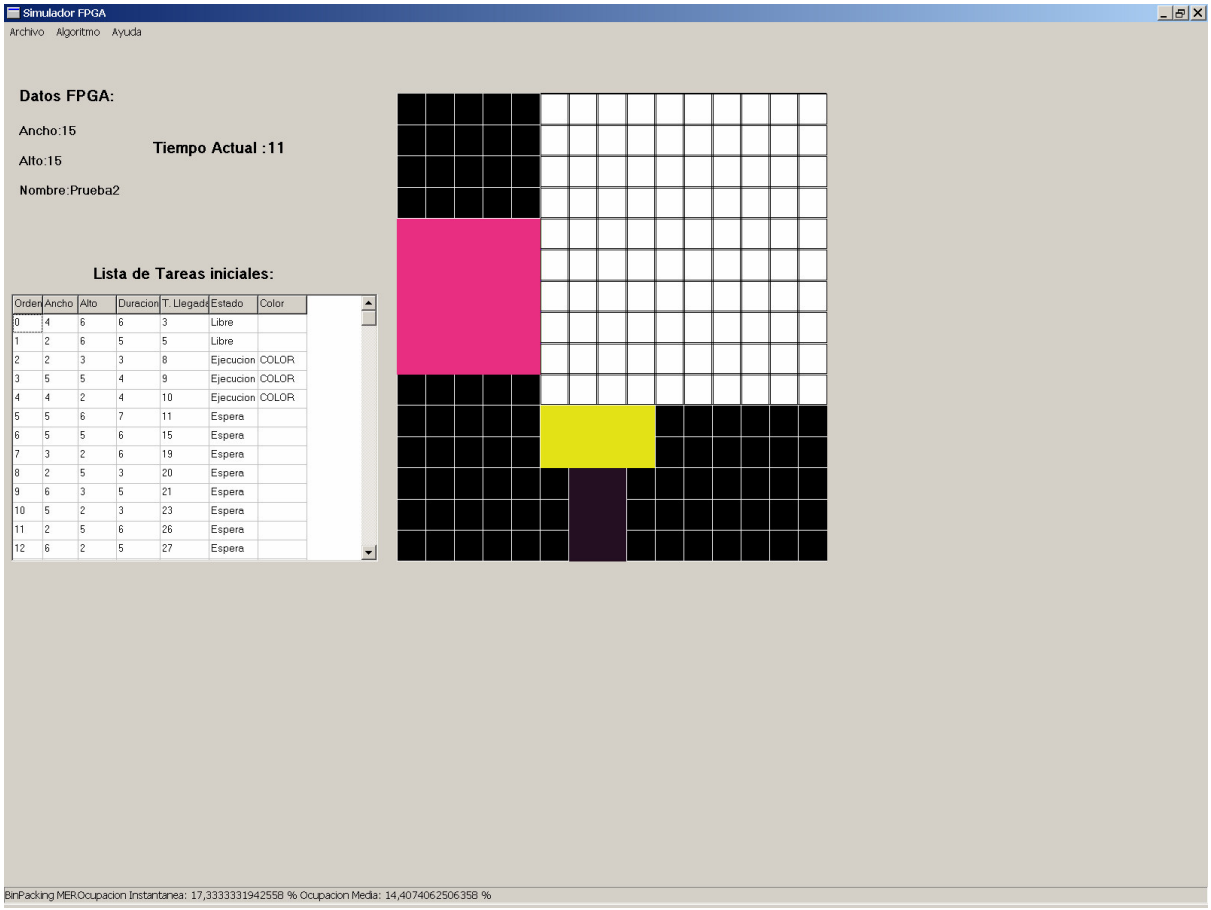
Ahora entra la tercera tarea en la FPGA (orden 2). Se sitúa en el rectángulo máximo anteriormente señalado.



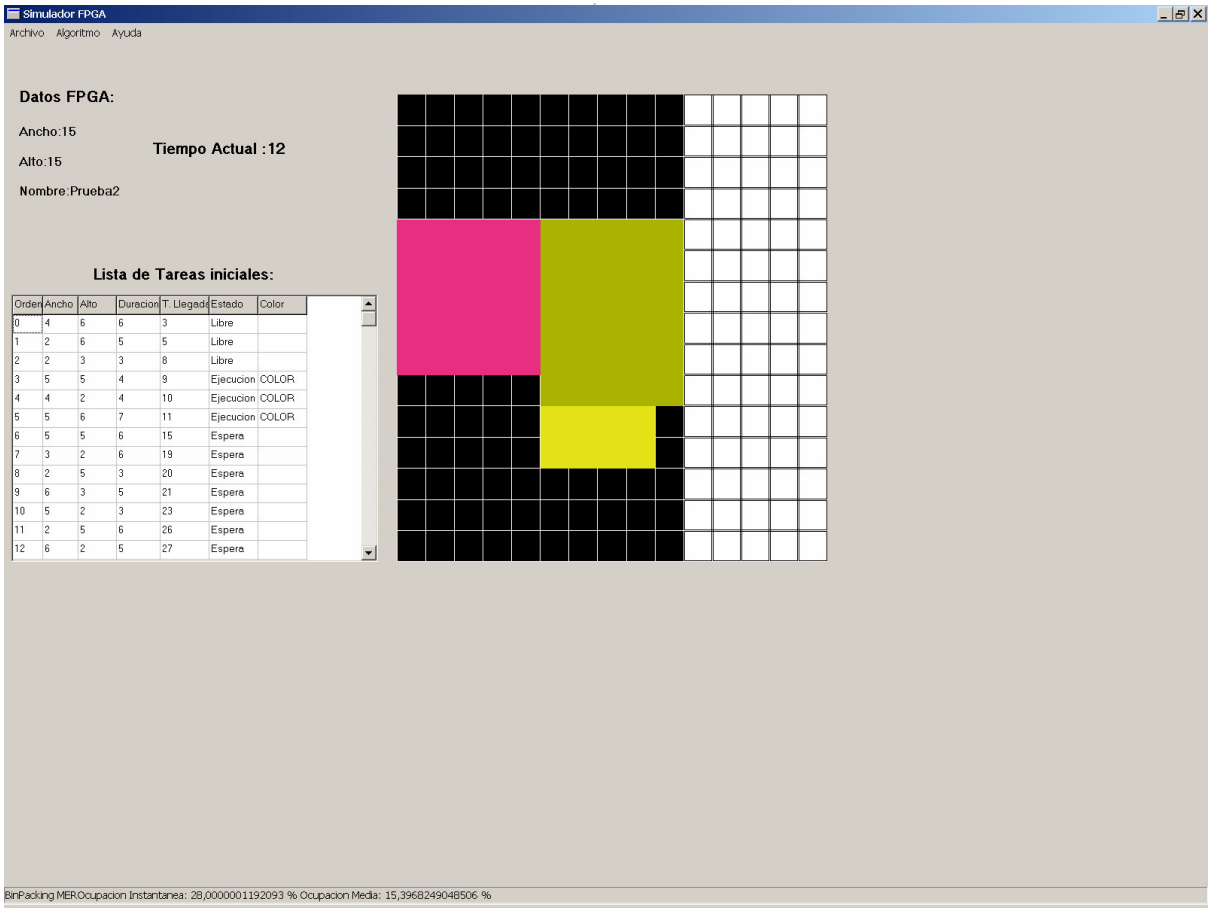
En el instante t=10 se dan dos situaciones distintas. En primer lugar sale la primera tarea y luego entra la cuarta tarea. En este caso esta última se sitúa en el rectángulo máximo anteriormente señalado ya que la salida de la primera tarea no provoca la generación de rectángulos vacíos mayores.



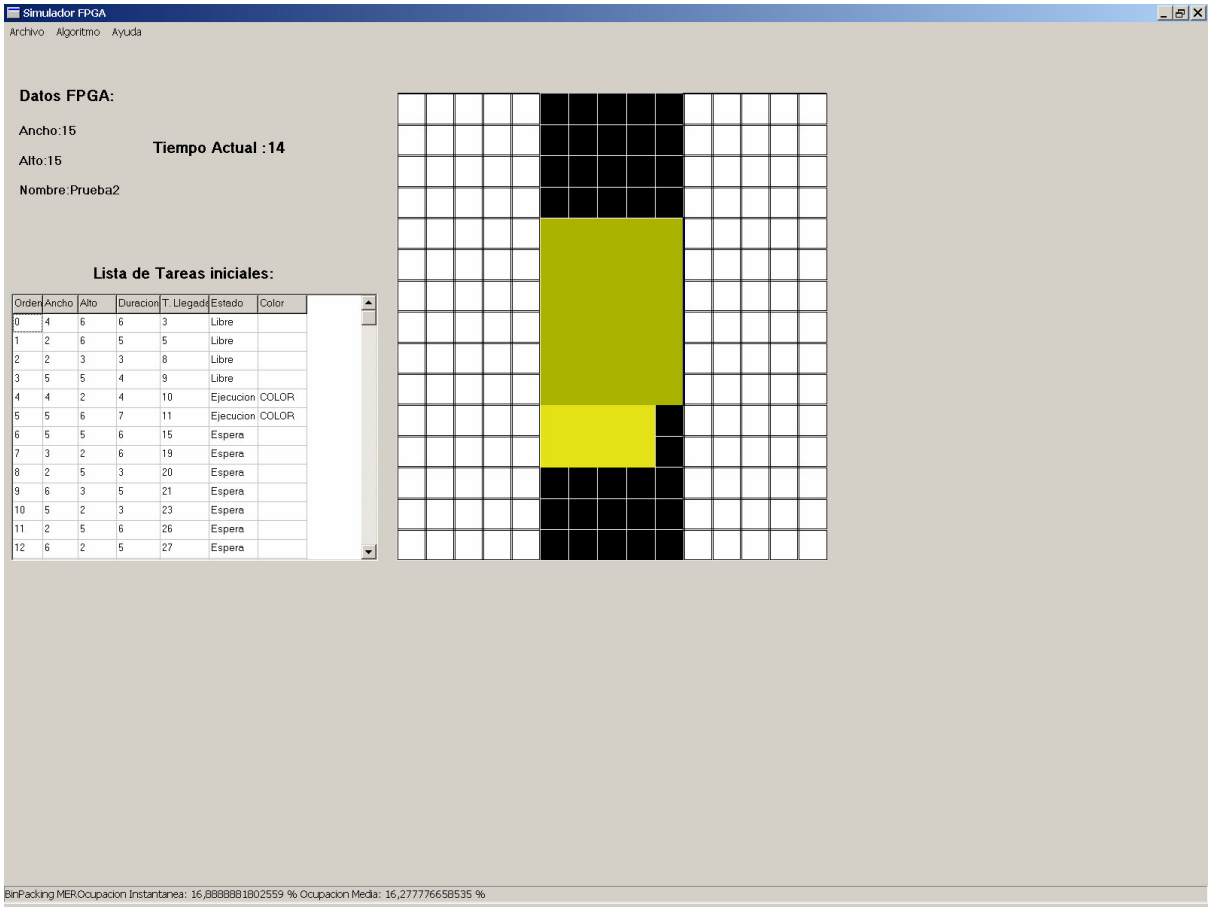
De nuevo sale de la FPGA la segunda tarea y entra una nueva tarea (orden 4). Se sitúa en el rectángulo máximo que genera la expiración de la segunda tarea.



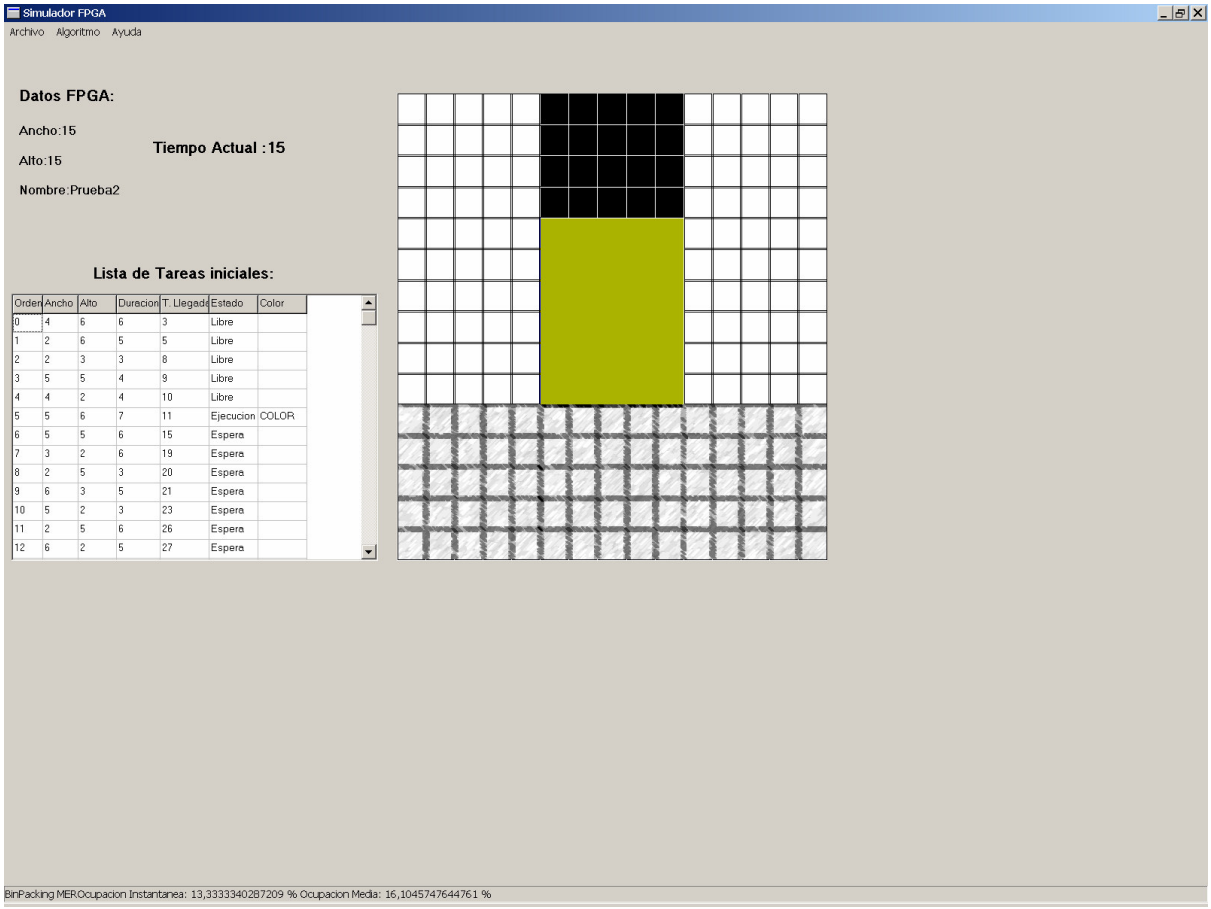
Se repite de nuevo la secuencia. Sale la tercera tarea (orden 2) y entra la sexta tarea (orden 5). Se coloca en el rectángulo máximo que aparece en la figura anterior ya que la expiración de la tercera tarea no da a lugar a un rectángulo de mayor área.



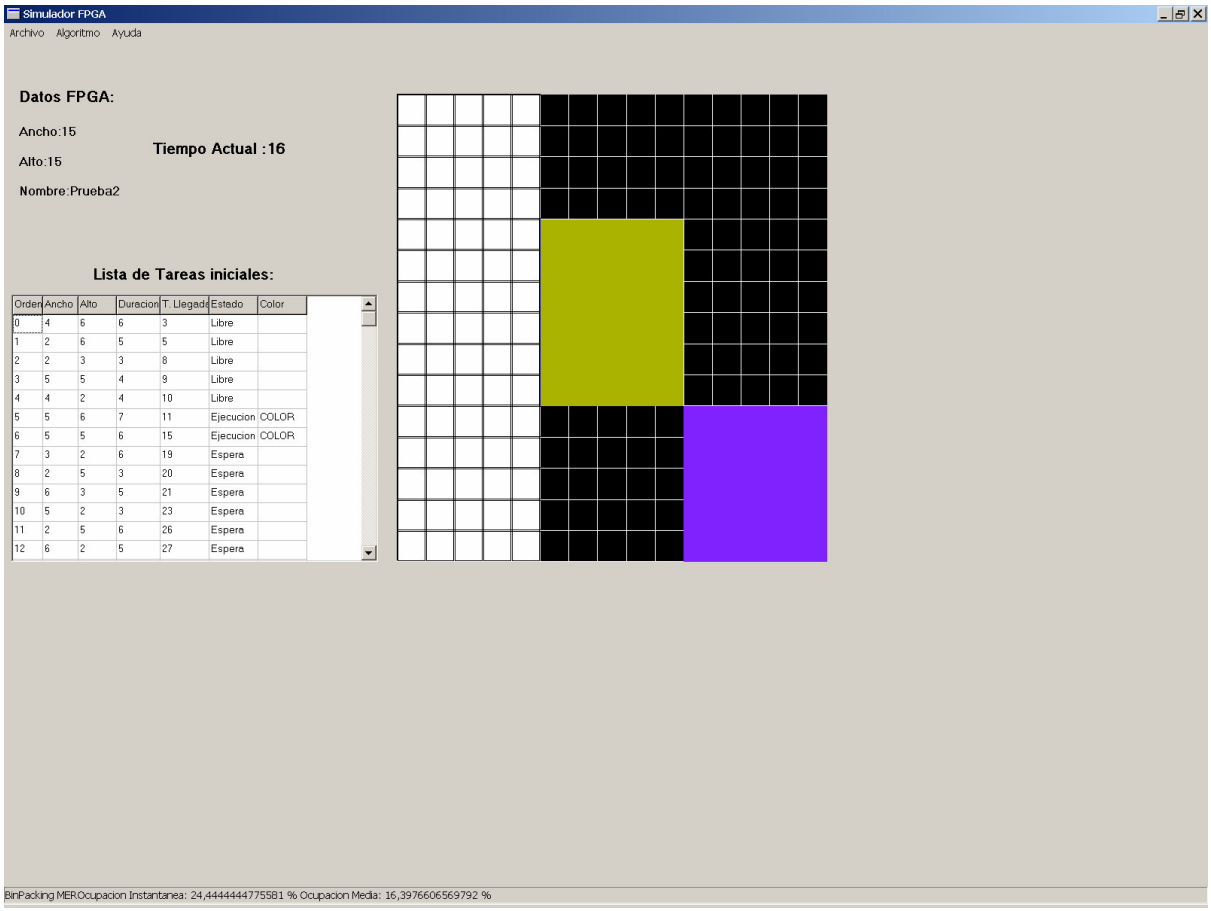
Simplemente sale la tarea de orden 3 de la FPGA. Esto provoca la aparición de 2 rectángulos máximos exactamente iguales a los lados.



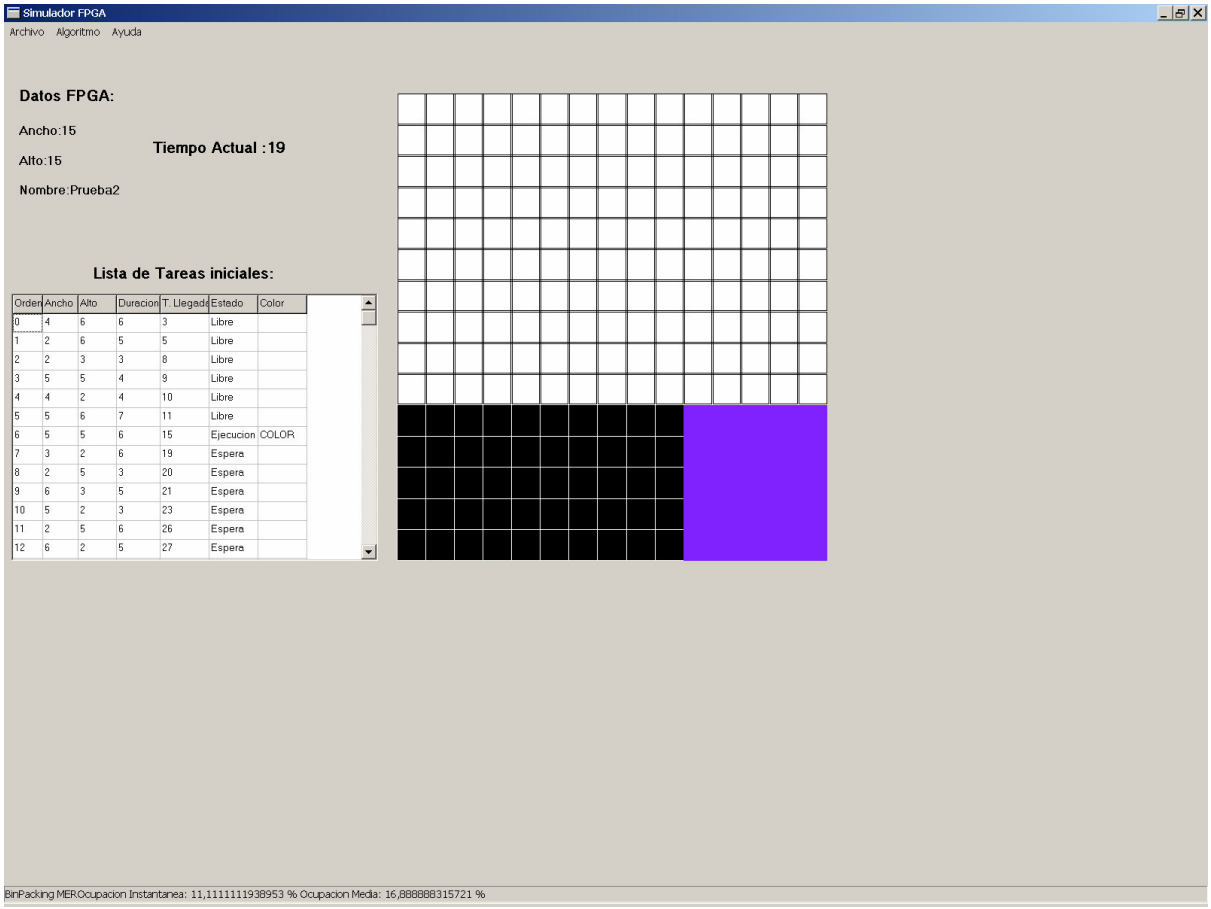
Sale la tarea de orden 4 de la FPGA y esto lleva a la situación de generarse 3 rectángulos máximos exactamente iguales.



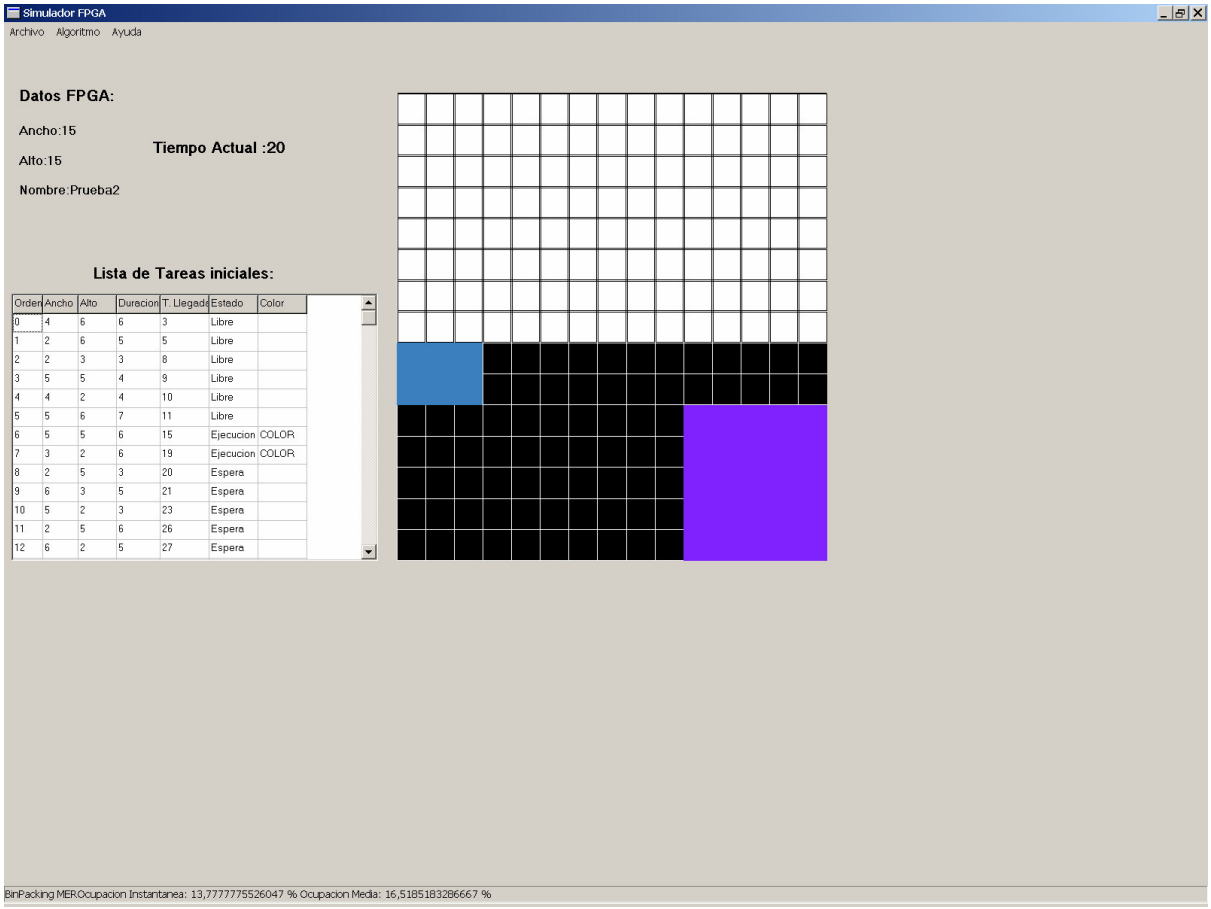
Entra la tarea de orden 6 en la FPGA en cualquiera de los rectángulos máximos anteriormente marcados.



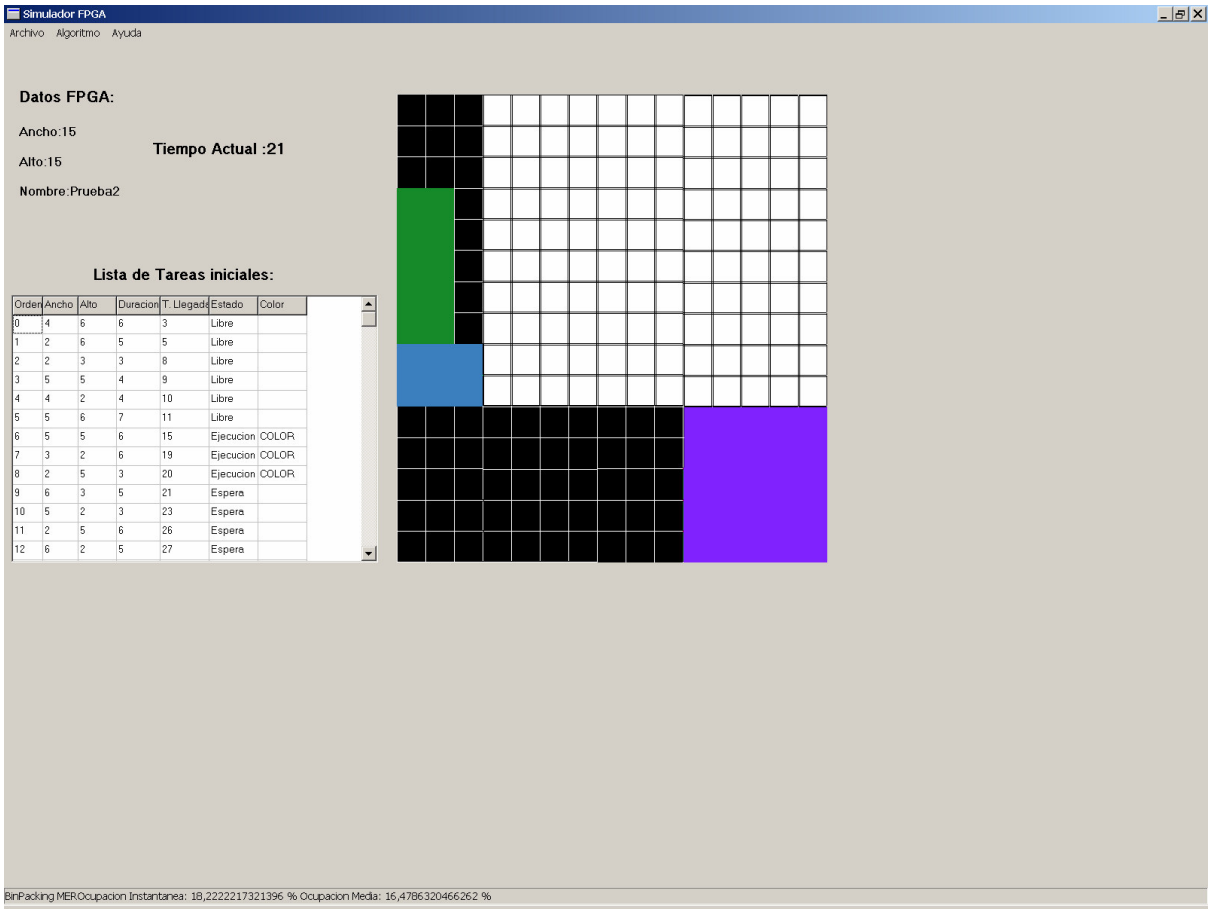
Sale tarea de orden 5 y solamente se queda la tarea de orden 6 en ejecución.



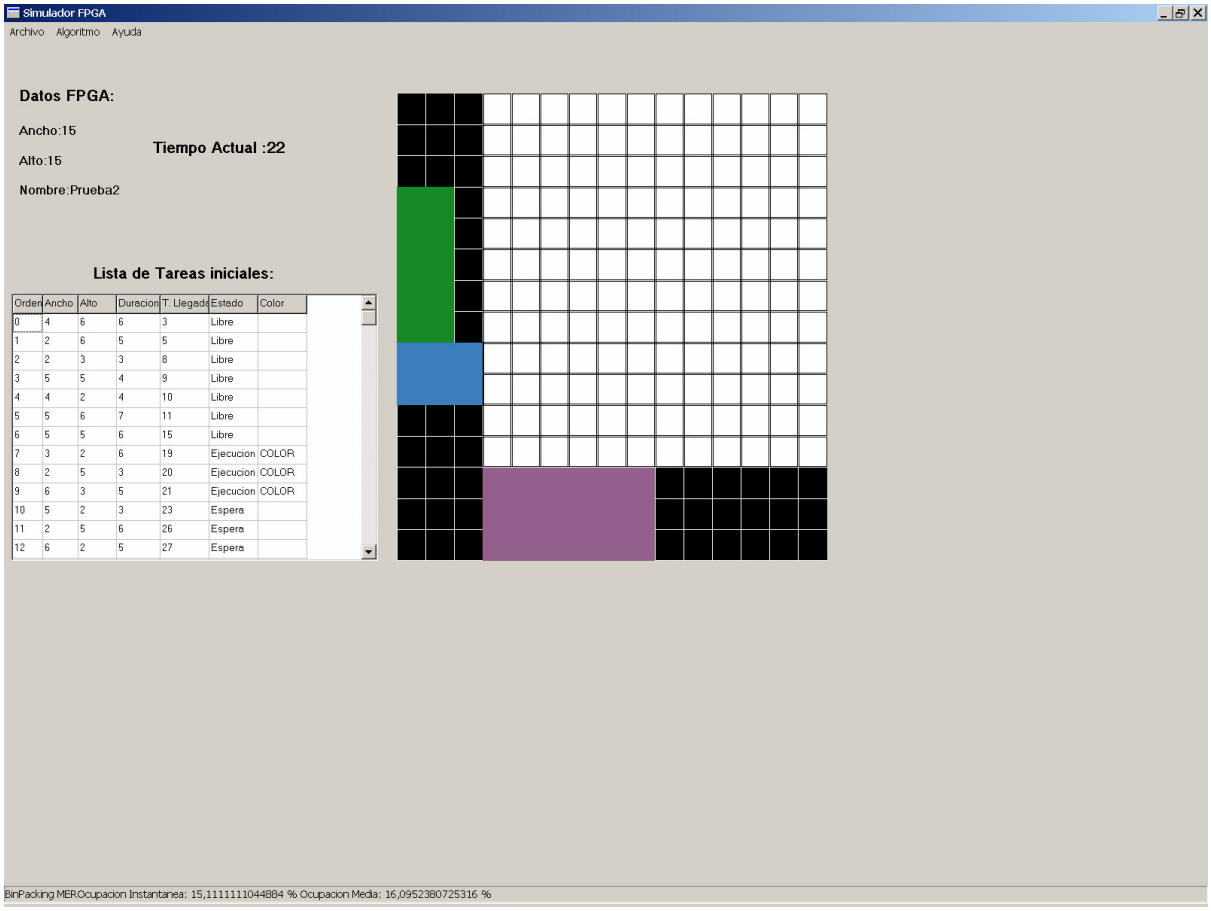
Entra la tarea de orden 7 situándose en el rectángulo máximo anteriormente marcado.


Simulador FPGA
Archivo Algoritmo Ayuda
Datos FPGA:
Ancho:15
Alto:15
Nombre:Prueba2
Tiempo Actual :20
Lista de Tareas iniciales:
Orden Ancho Alto Duracion T. Llegada Estado Color
0 4 6 6 3 Libre
1 2 6 5 5 Libre
2 2 3 3 8 Libre
3 5 5 4 9 Libre
4 4 2 4 10 Libre
5 5 6 7 11 Libre
6 5 5 6 15 Ejecucion COLOR
7 3 2 6 19 Ejecucion COLOR
8 2 5 3 20 Espera
9 6 3 5 21 Espera
10 5 2 3 23 Espera
11 2 5 6 26 Espera
12 6 2 5 27 Espera
BinPacking MER Ocupacion Instantanea: 13,7777775526047 % Ocupacion Media: 16,5185183286667 %

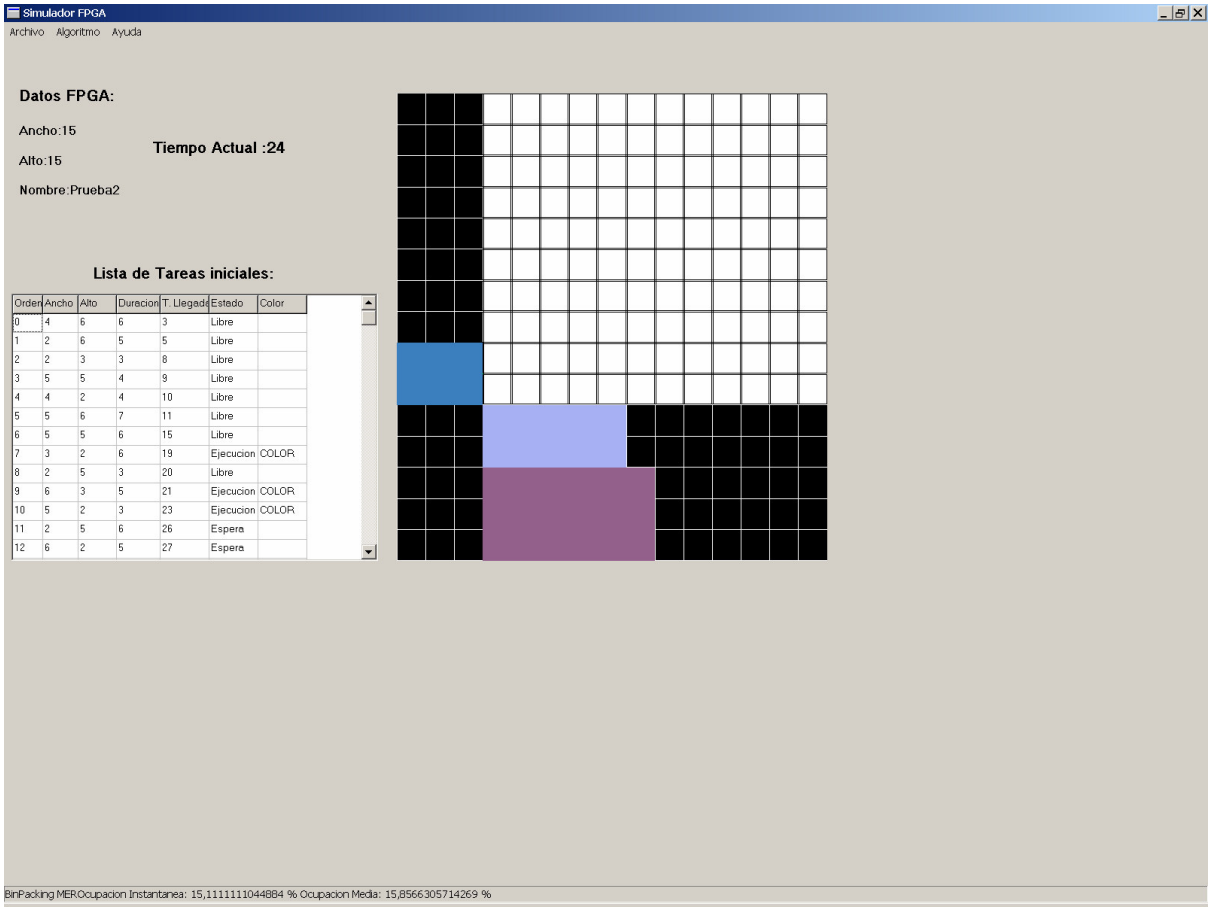
Entra una nueva tarea (orden 8) y aparece sombreado el rectángulo máximo. Tenemos que señalar un aspecto importante del algoritmo. Si llega una tarea y no cabe en la FPGA se incrementan en 1 unidad de tiempo todos los tiempos de llegada de las tareas en espera.



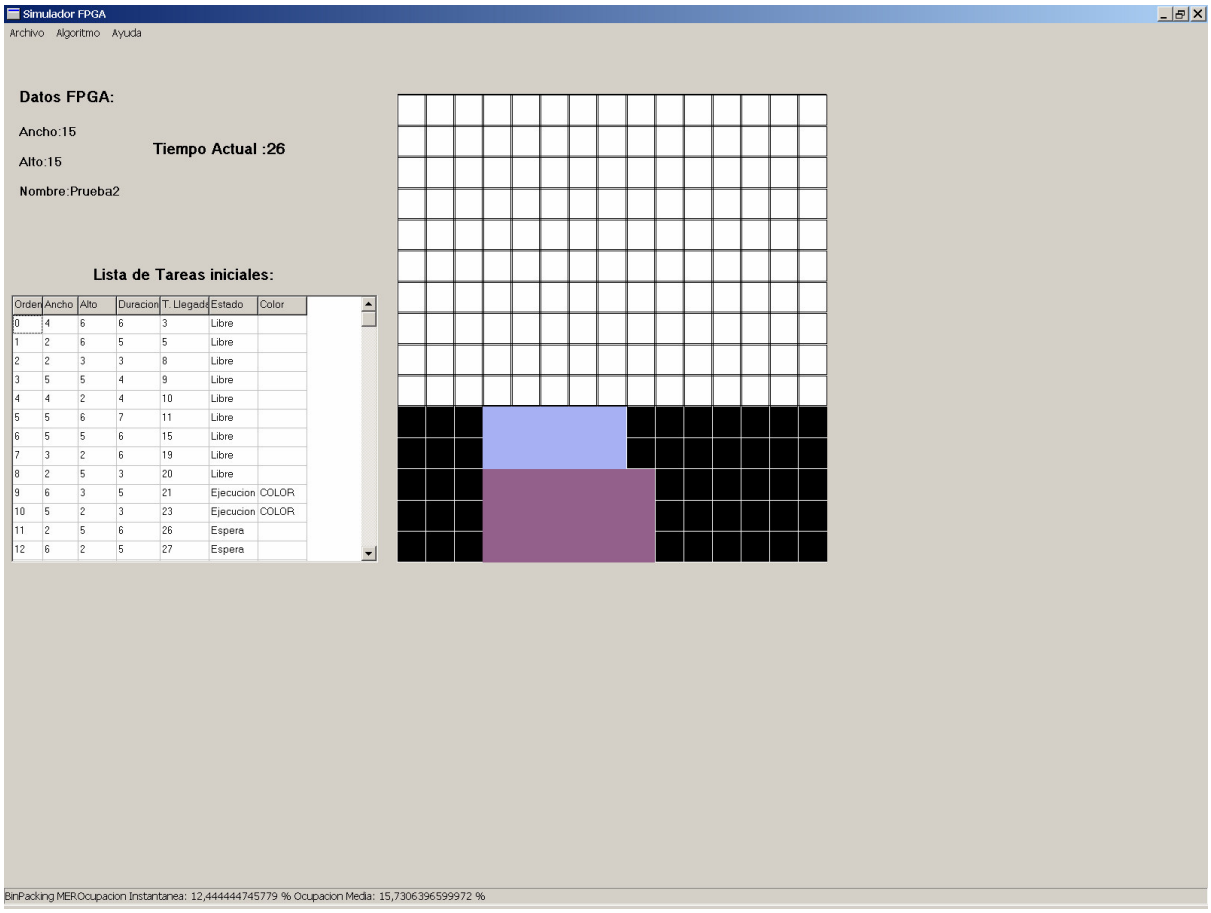
Sale la tarea de orden 6 y entra la tarea de orden 9. Se sitúa en el rectángulo máximo generado a partir de la expiración de la tarea de orden 6.



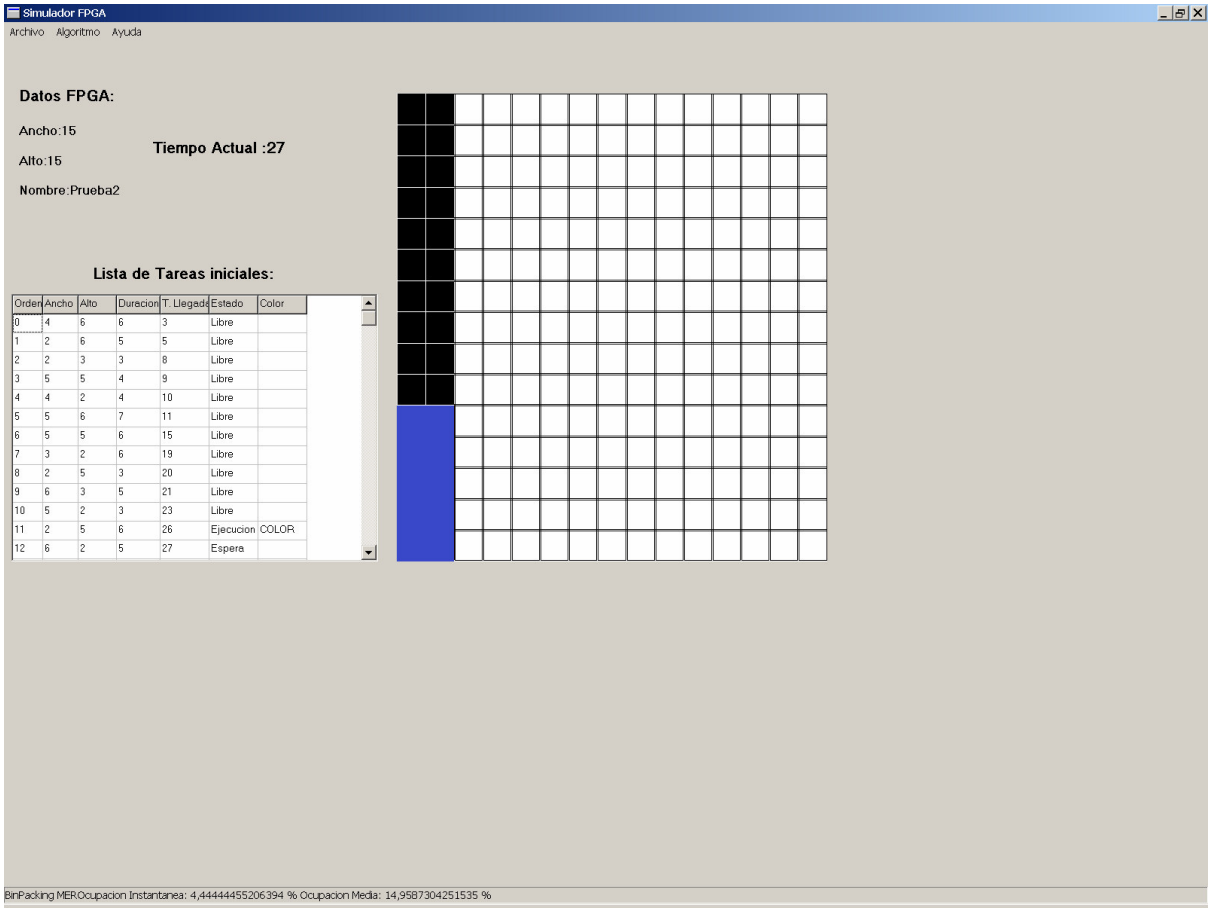
Ahora sale la tarea de orden 8 y entra la tarea de orden 10. Esta última se sitúa en el rectángulo máximo anteriormente señalado.



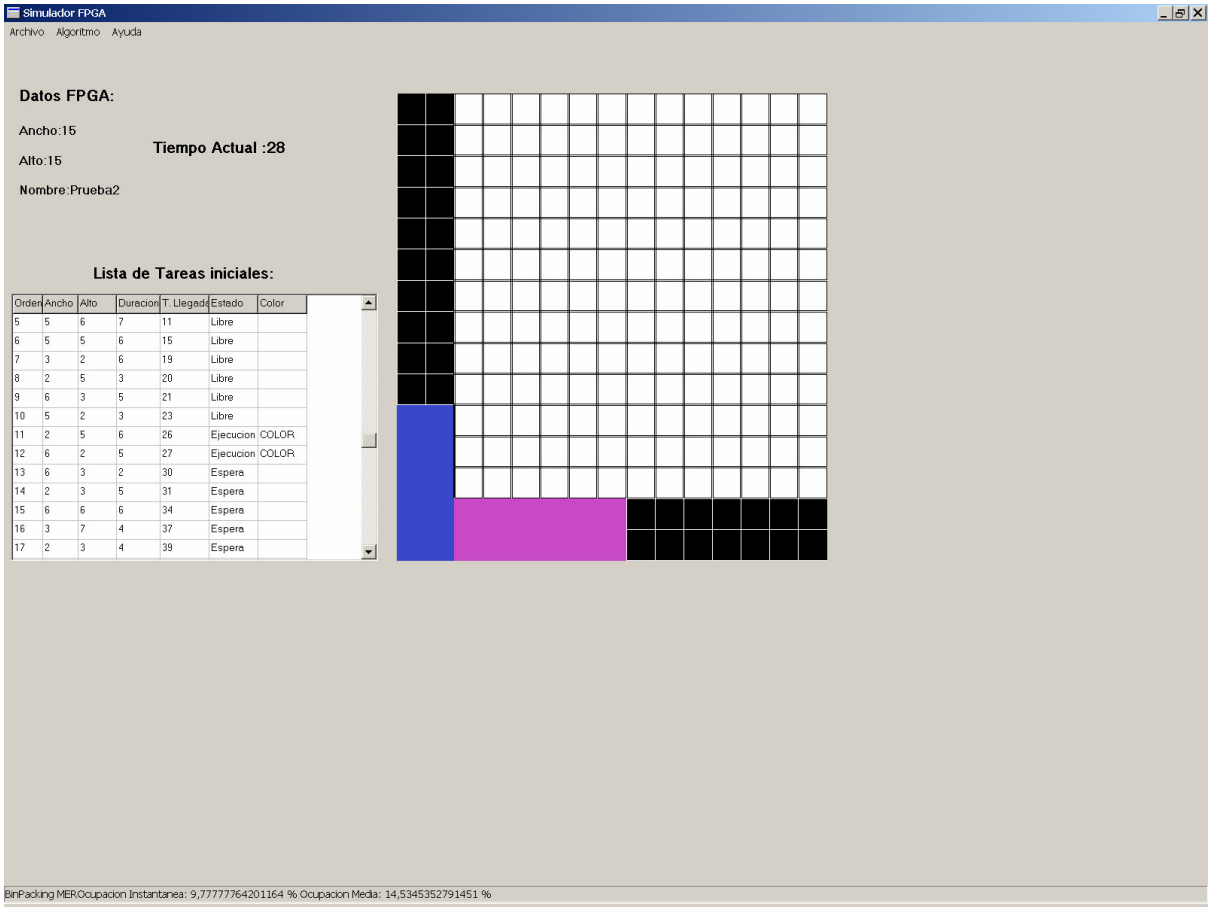
Sale la tarea de orden 7 y genera el rectángulo máximo señalado a continuación.



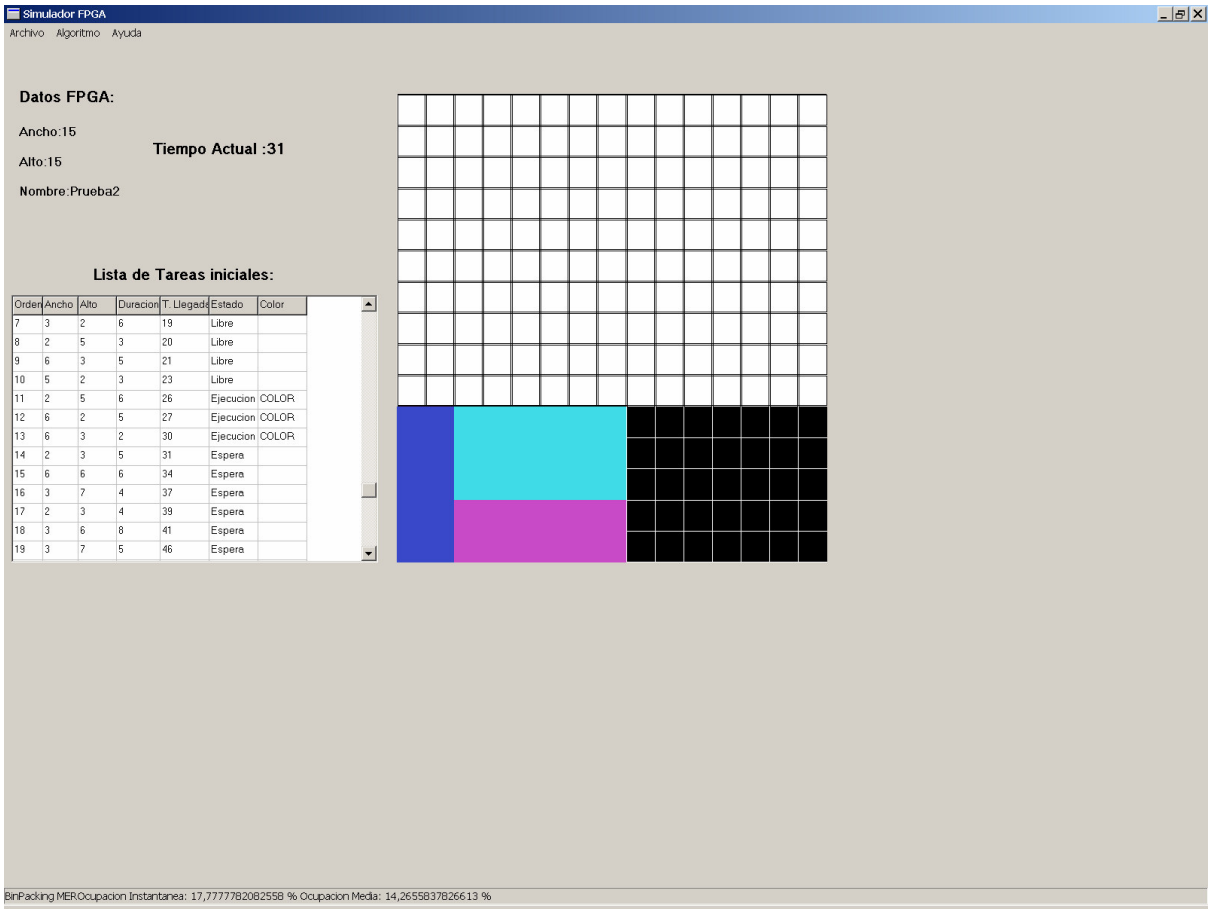
Salen las 2 tareas que estaban en ejecución y entra la tarea en una FPGA totalmente vacía. Entonces por defecto siempre se coloca en los vértices inferiores izquierdos como hemos dicho antes.



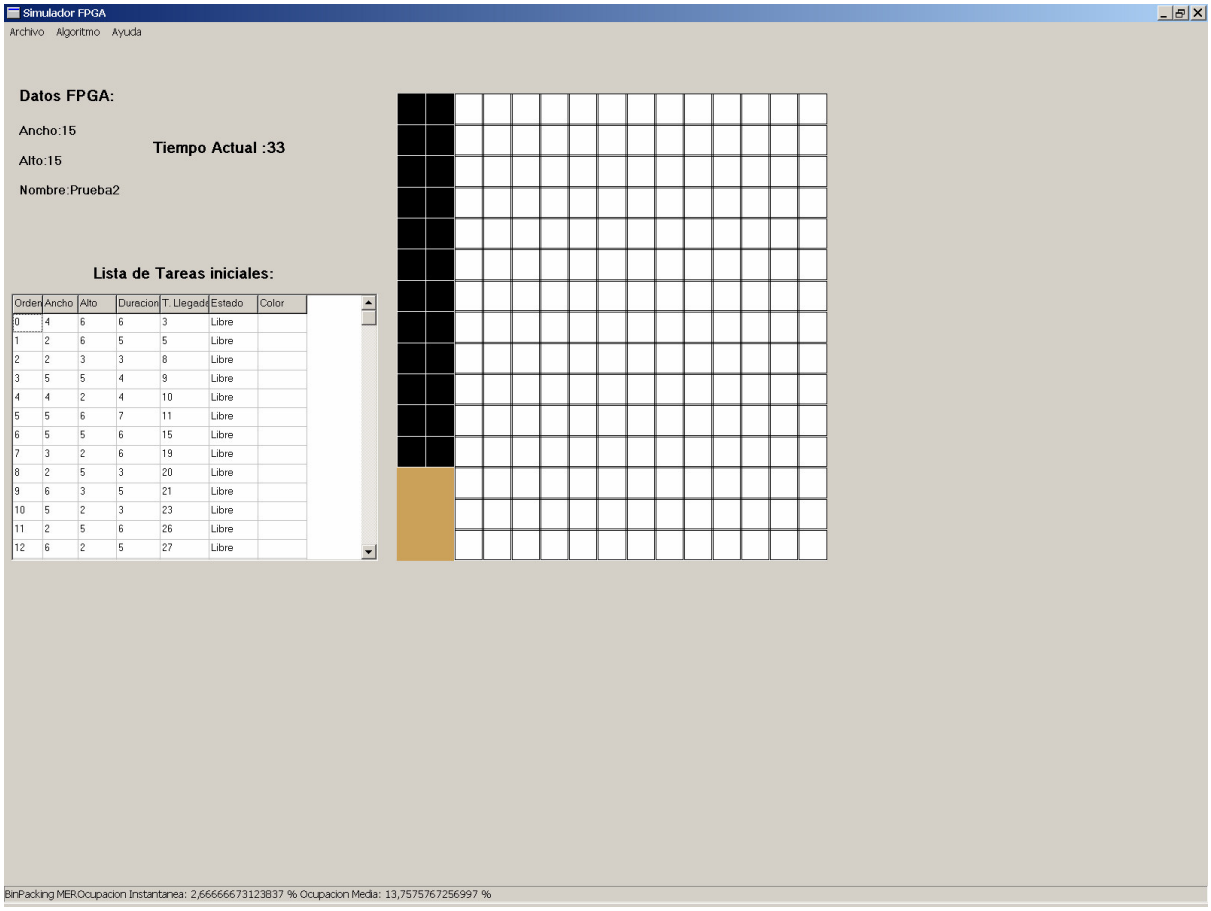
Entra una nueva tarea y se coloca en el rectángulo máximo que aparece en la figura anterior.



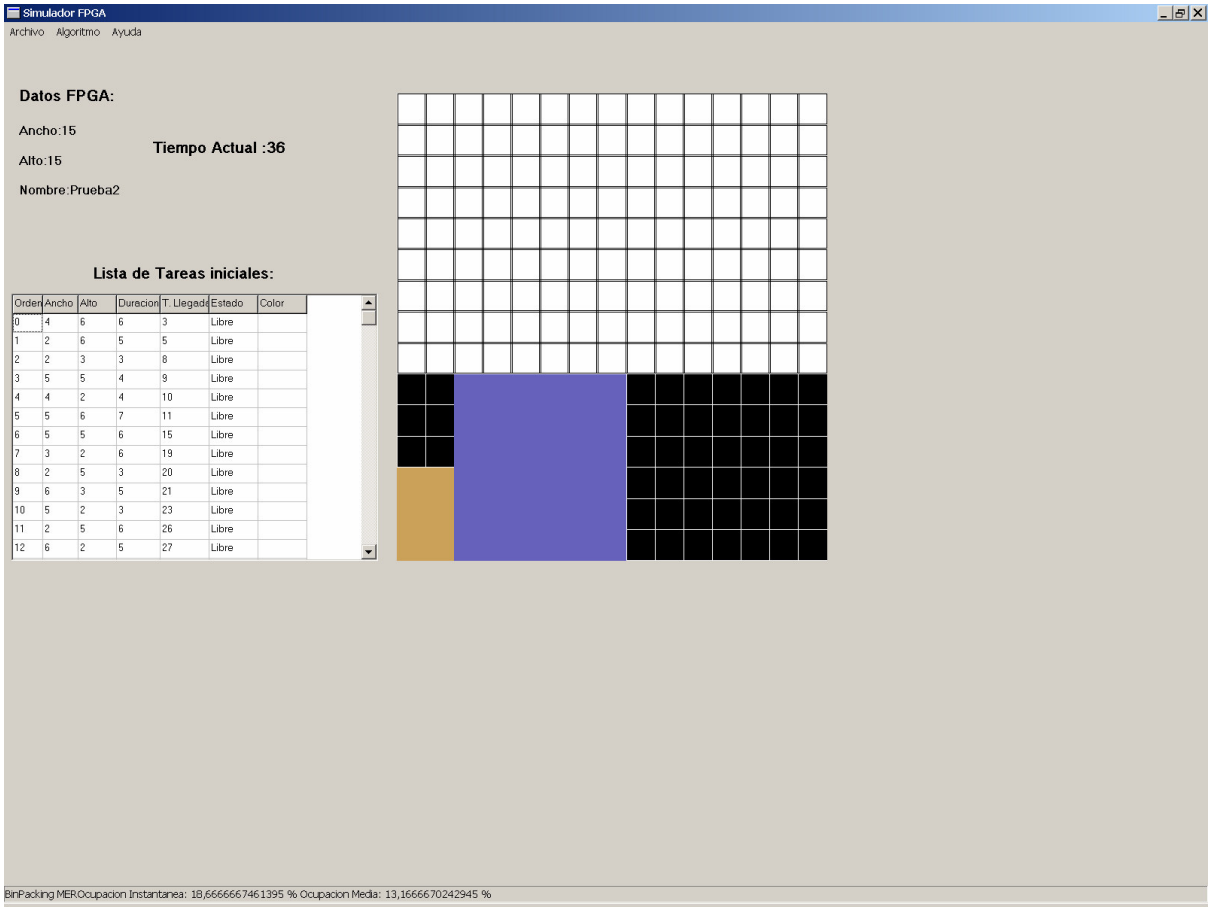
Entra una nueva tarea y se sitúa en el rectángulo máximo de la FPGA anterior.



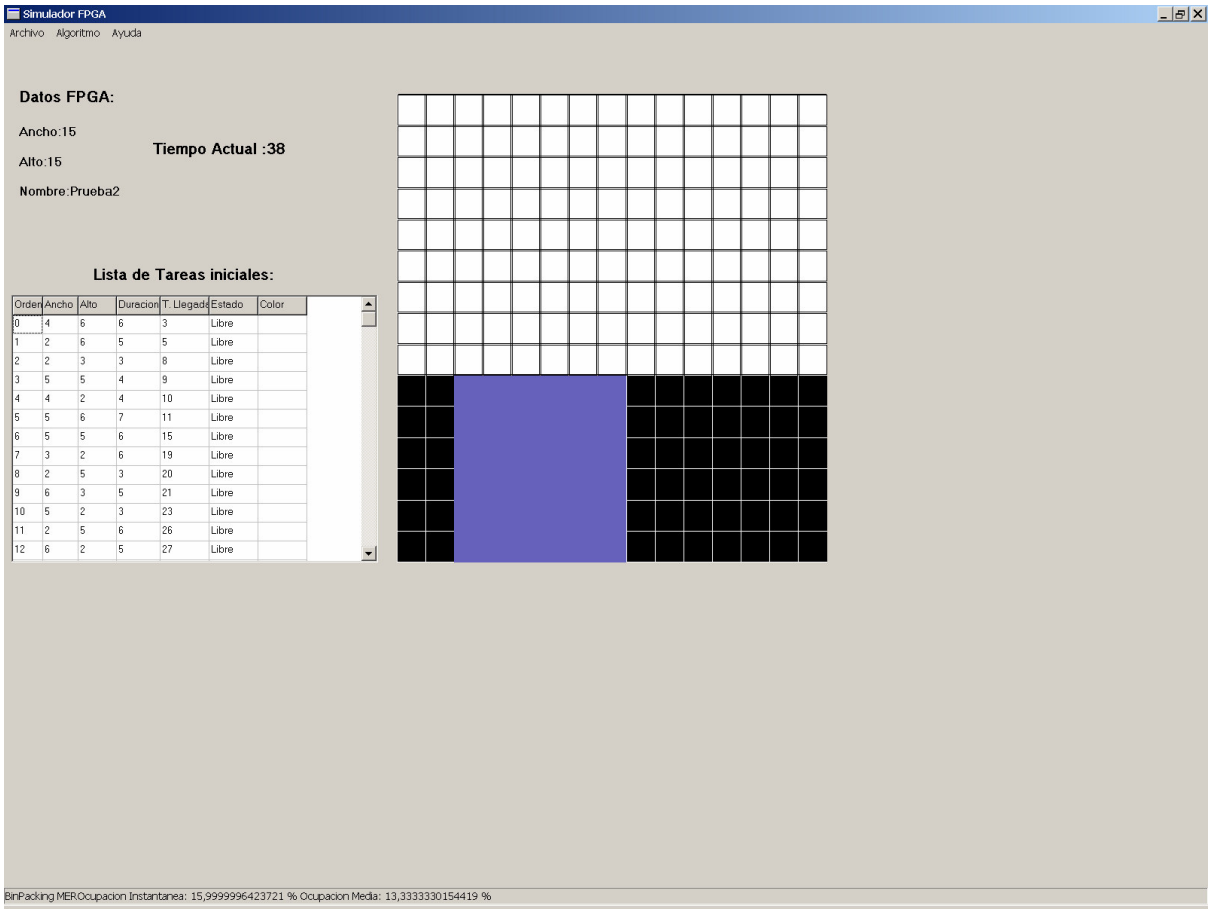
En el instante t=33 expiran las 3 tareas que estaban en ejecución y entra una nueva tarea con la FPGA totalmente vacía.



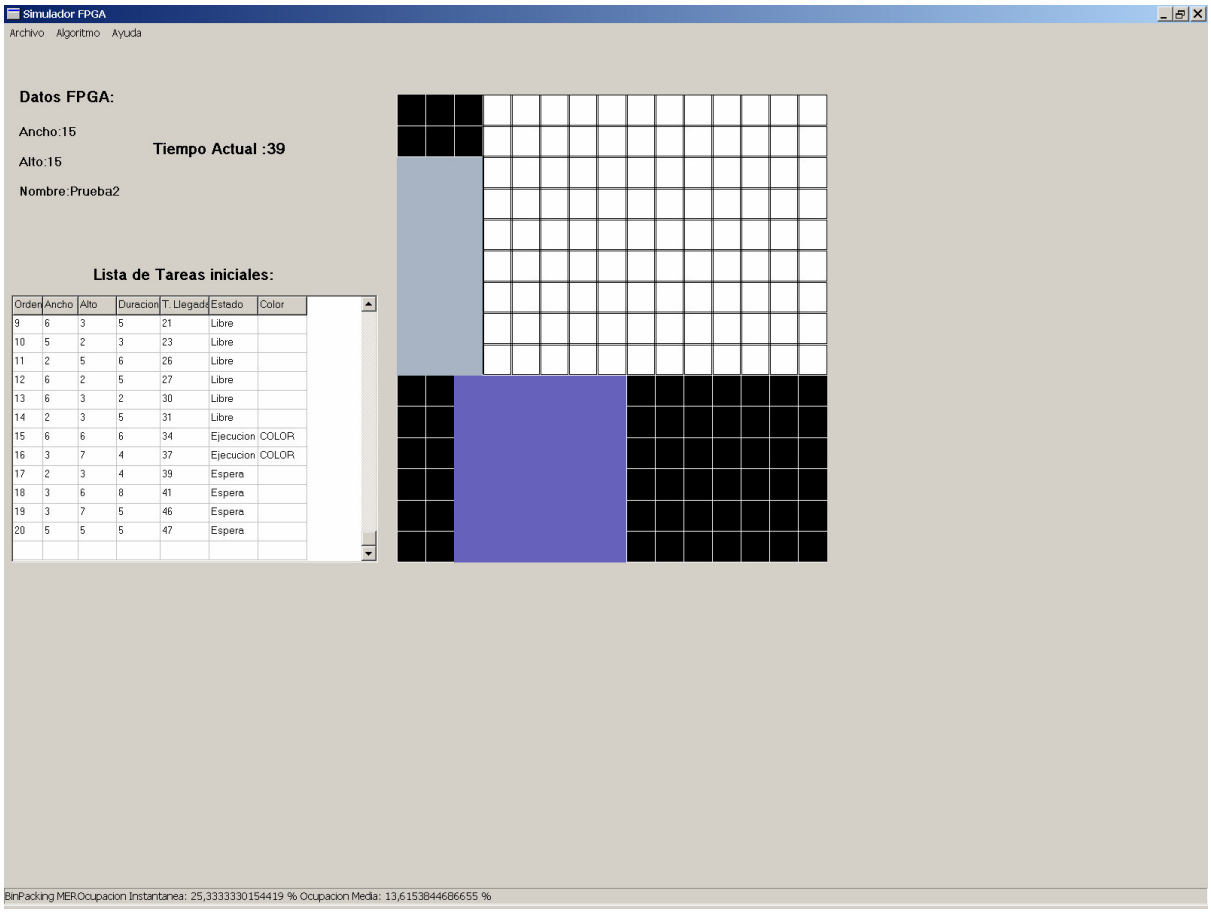
Entra una nueva tarea en la FPGA.



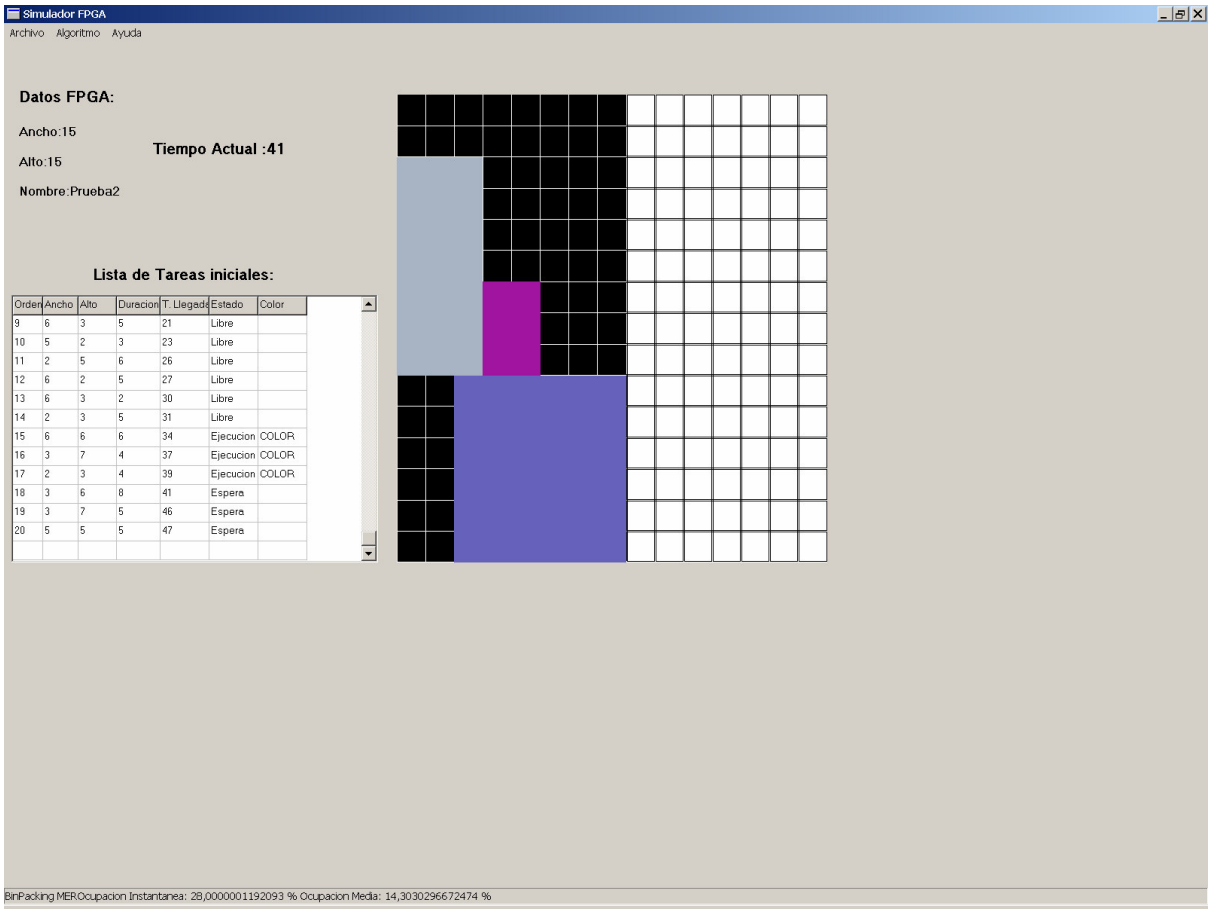
Sale una tarea de la FPGA. Volvemos a situaciones explicadas anteriormente explicadas.



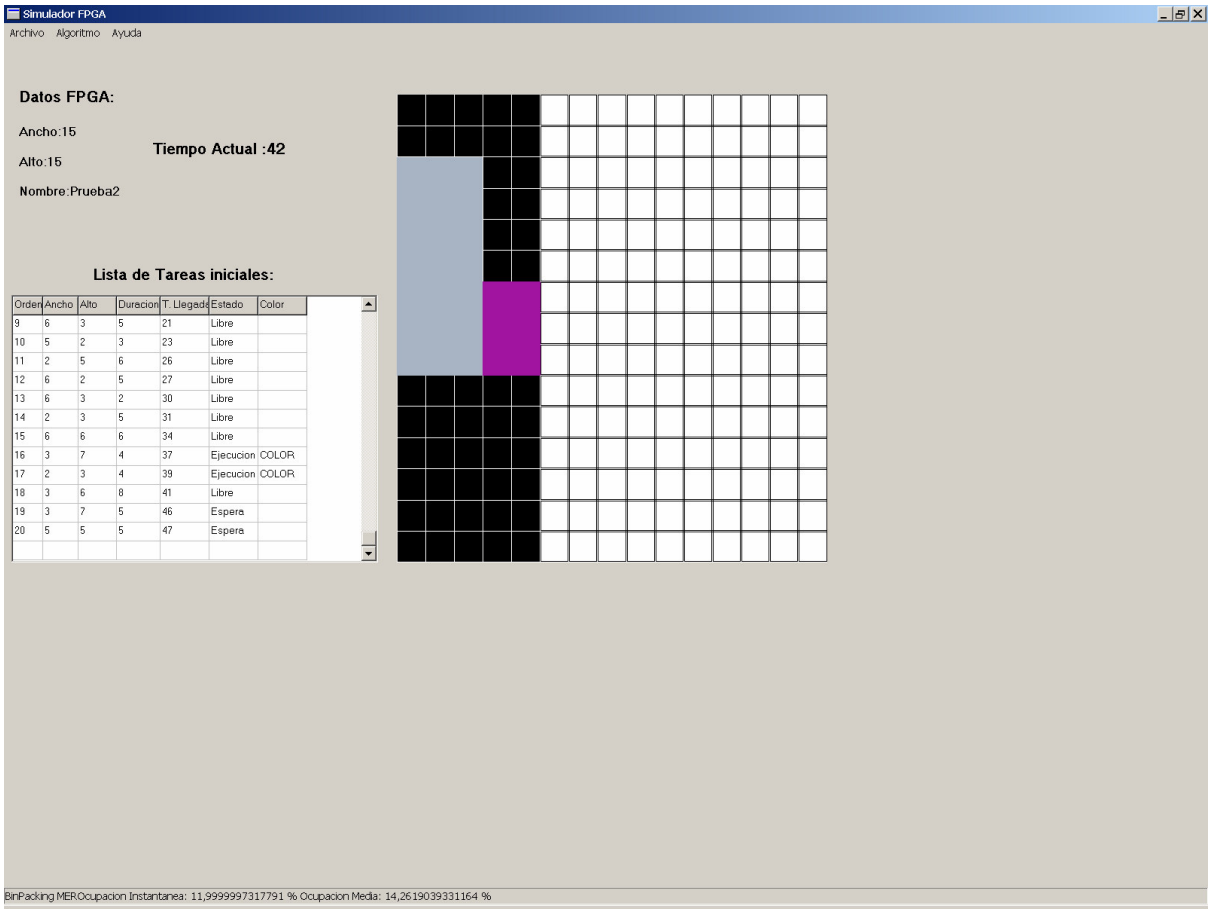
Entra una tarea en la FPGA. El rectángulo de mayor área es el que aparece sombreado de blanco (12 x 9 = 108) ya que el rectángulo situado en la vertical derecha es de 7 x 15 = 105.



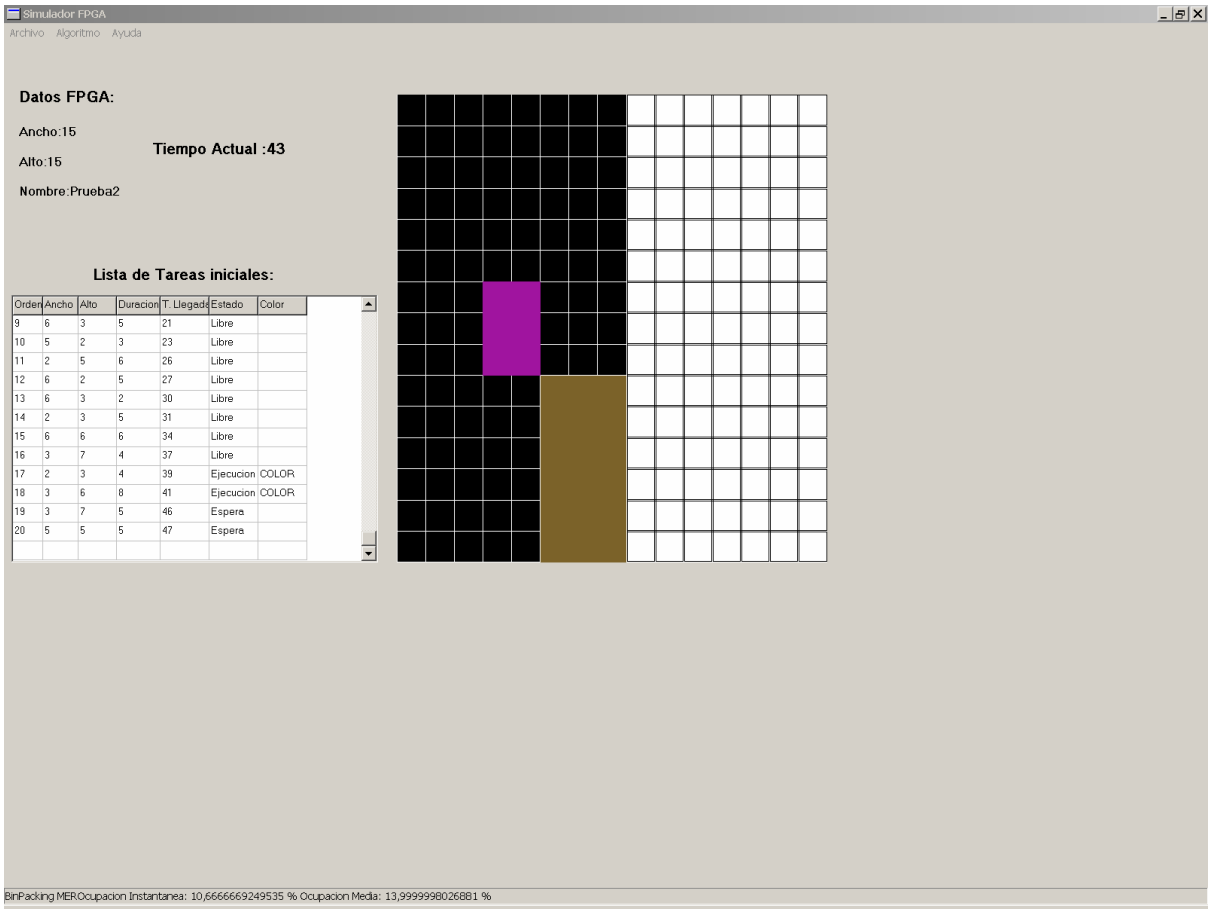
Entra la tarea de orden 18 en la FPGA.



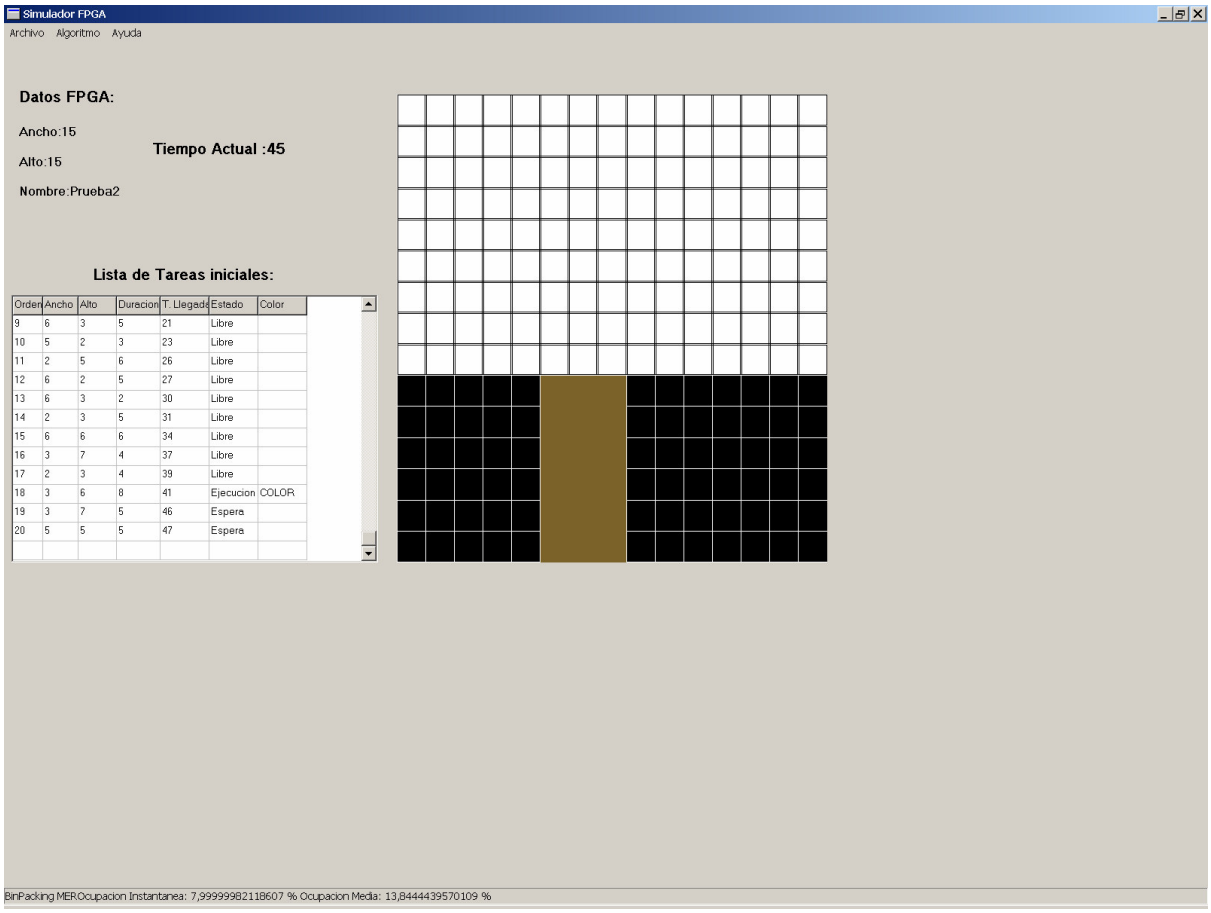
Sale una tarea de la FPGA.



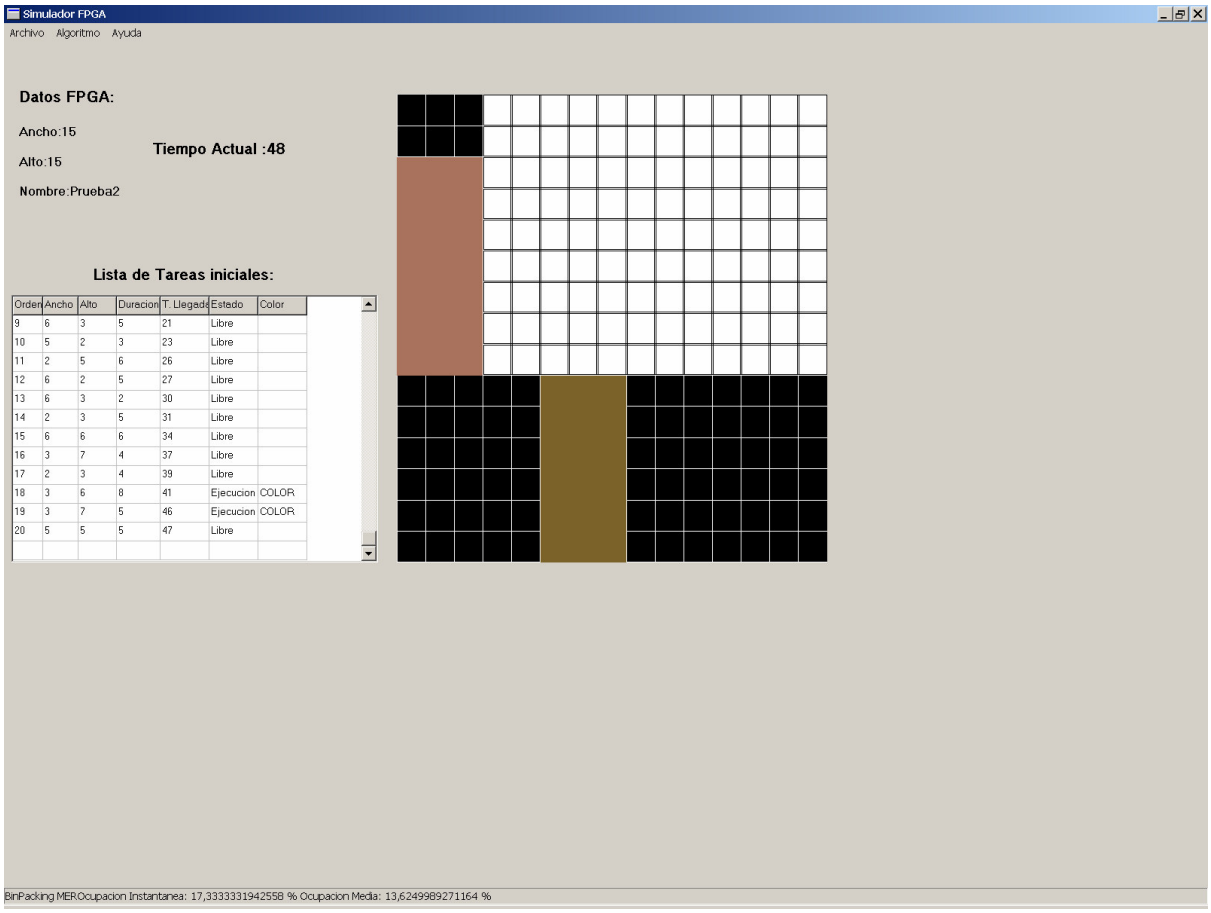
Sale y entra una tarea de la FPGA.



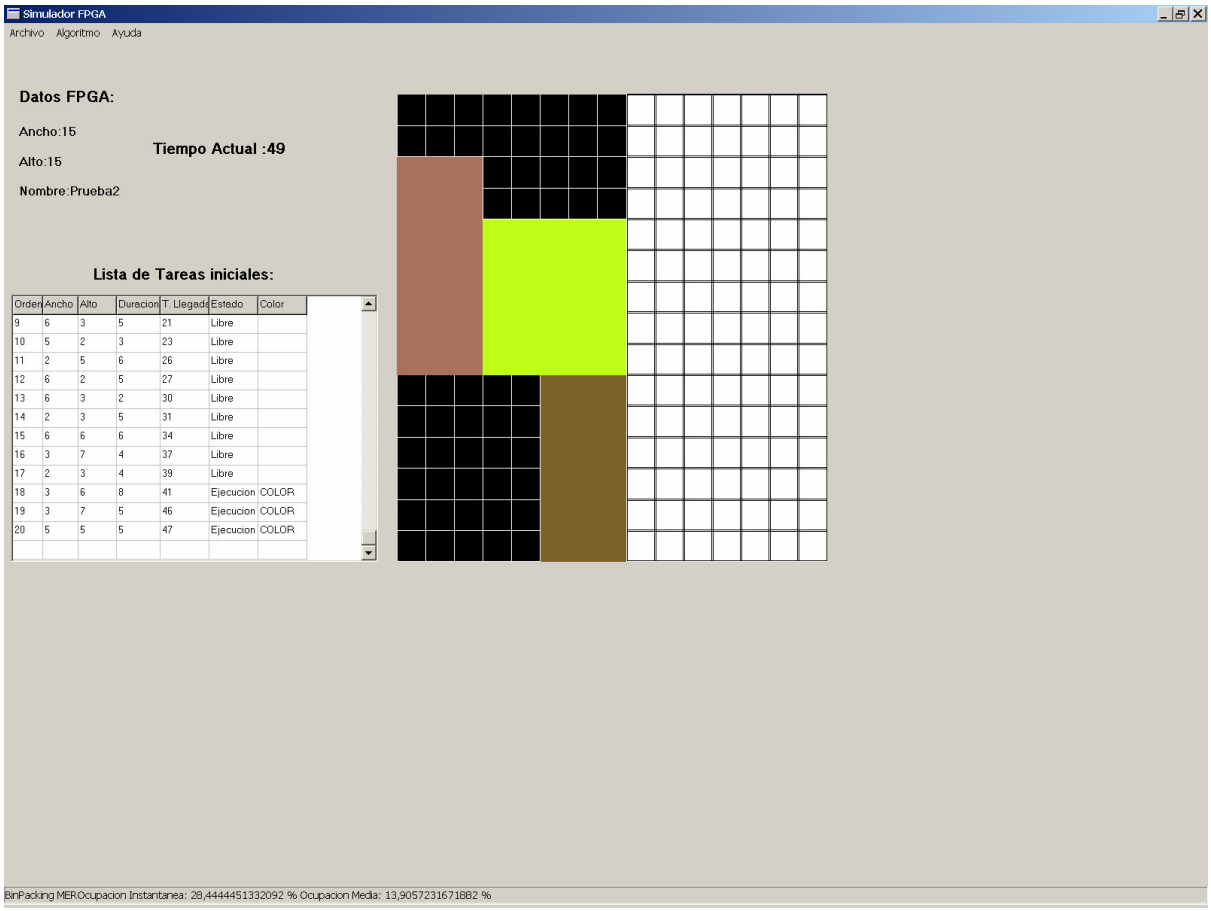
Sale una tarea de la FPGA y el rectángulo máximo aparece sombreado a continuación:



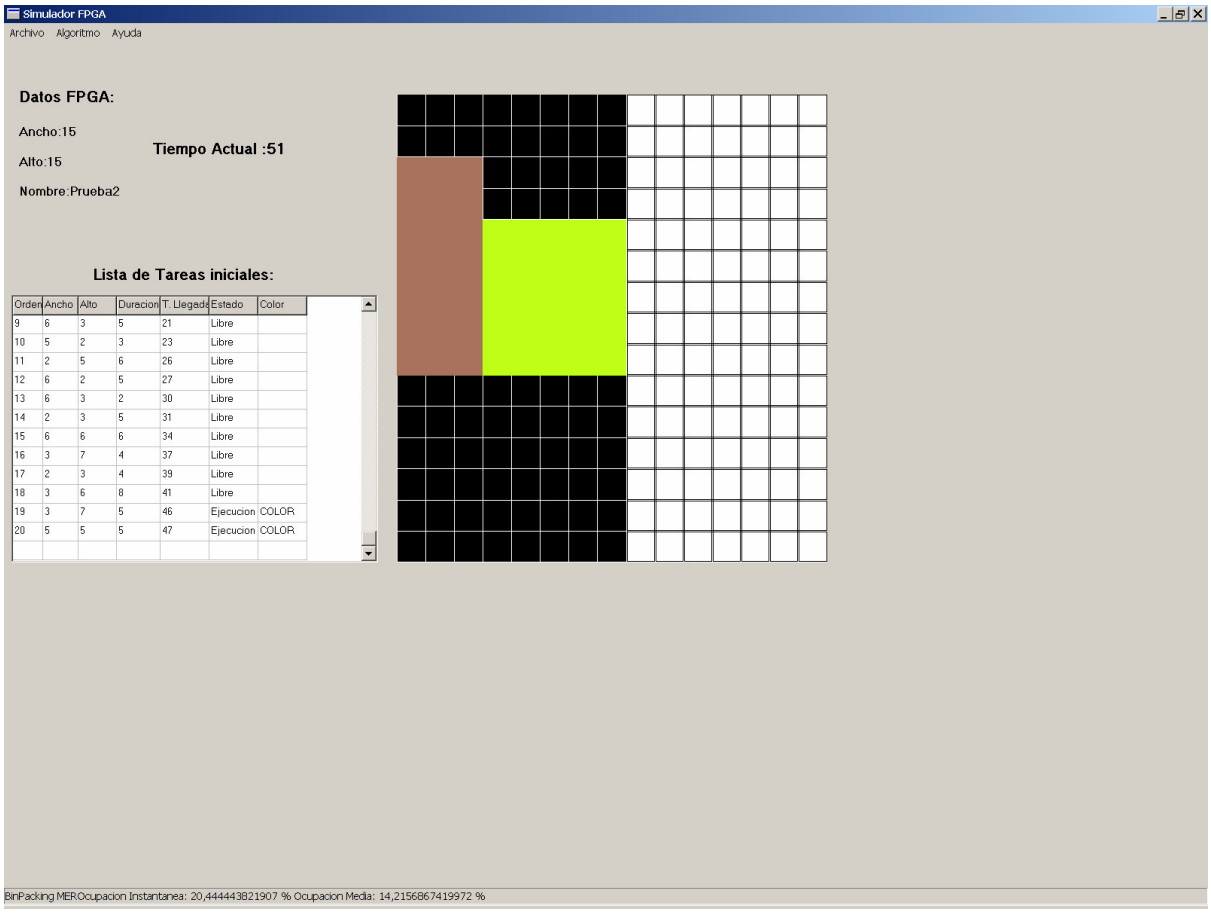
Entra una nueva tarea.



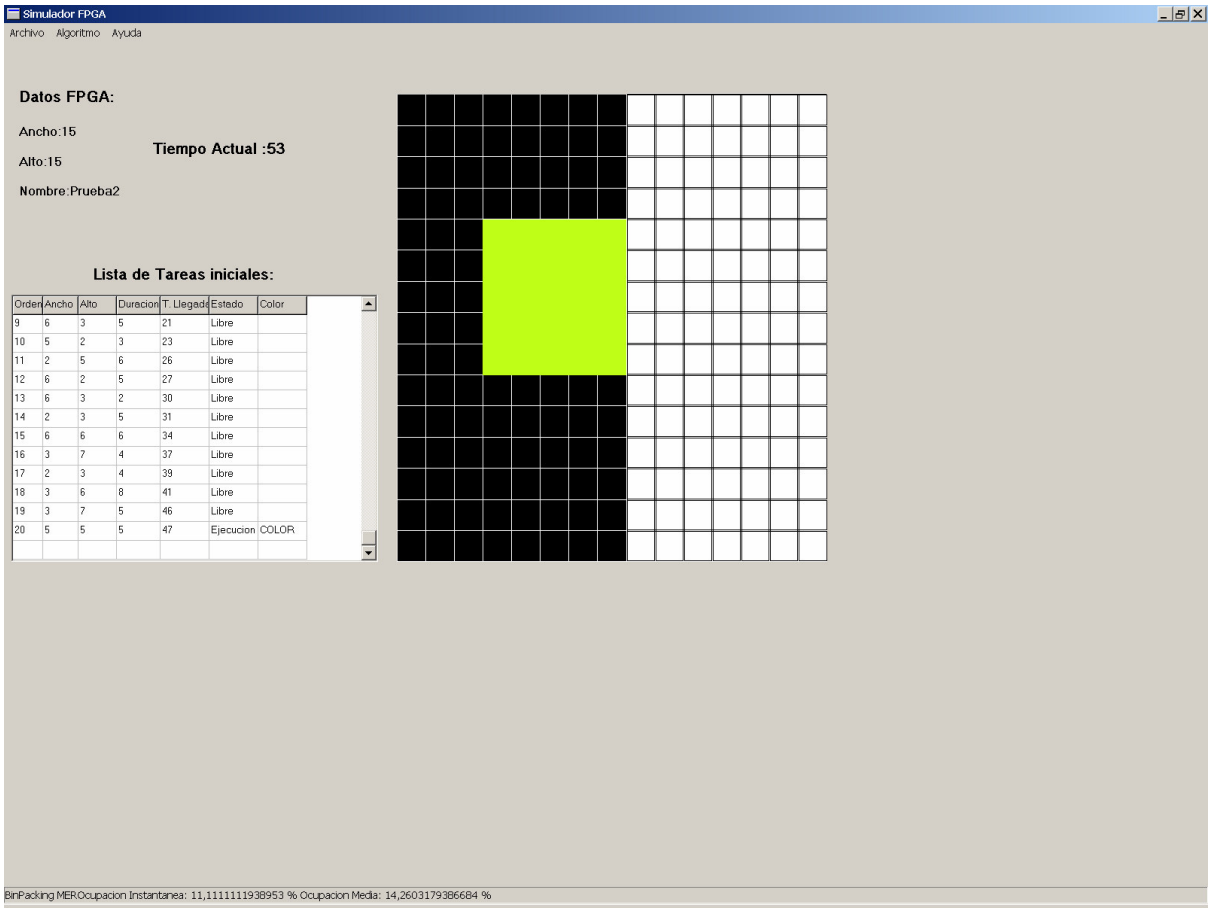
Entra una nueva tarea y se coloca en el rectángulo máximo de la figura anterior.



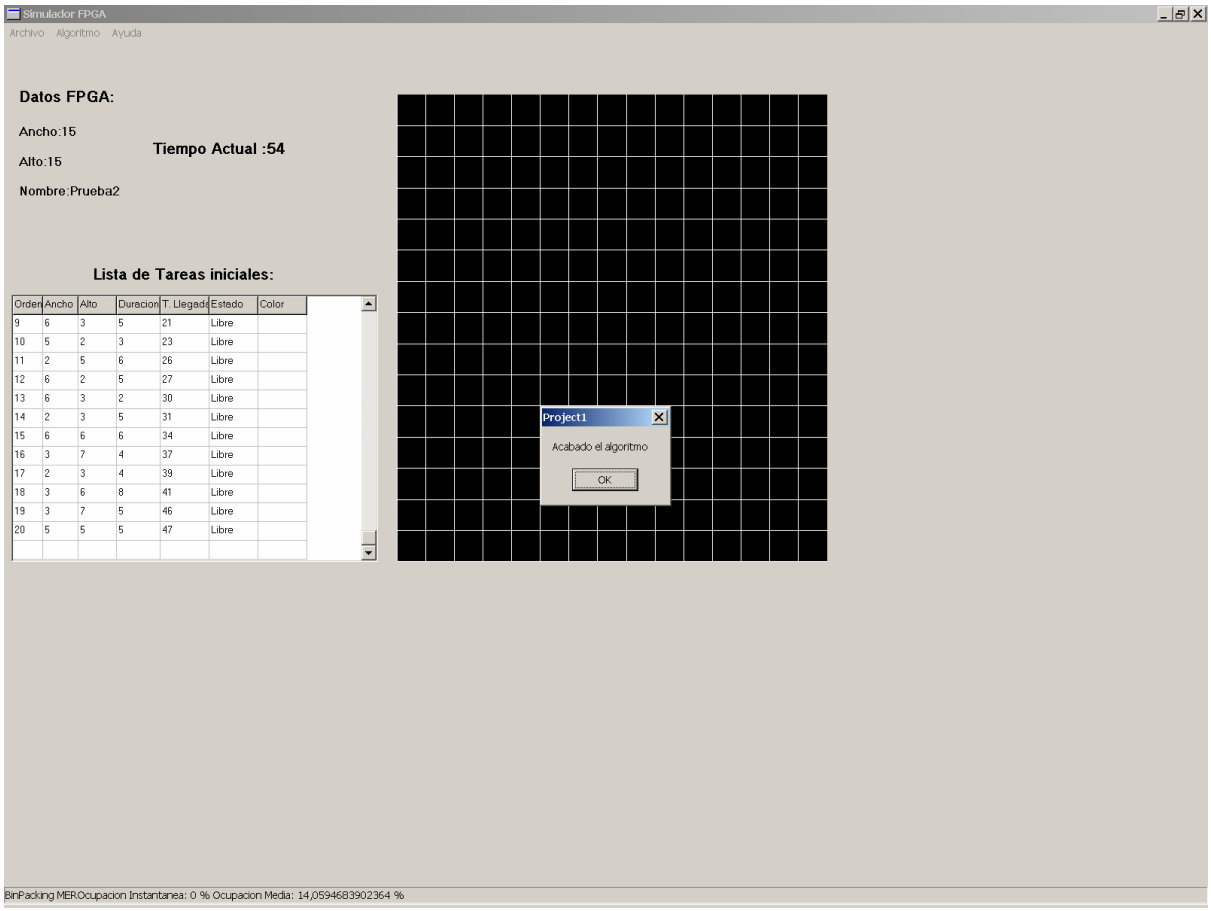
Sale una tarea de la FPGA. El rectángulo máximo es el de la derecha vertical (7 x 15) o el horizontal de abajo (15 x 7).



Sale una tarea de la FPGA.



Sale la última tarea de la FPGA y acaba el algoritmo.



Tecnología utilizada

El software y hardware empleado para la realización del empleado se resume en:

- Máquinas con procesador Pentium IV.
- OpenGL.
- Borland C++ Builder.
- Internet Explorer para búsquedas de información.

Batería de pruebas

Se va a realizar una serie de simulaciones con distintos tamaños de lista de tareas y FPGA´s. Veremos como varía un dato estadístico importante como es la ocupación media.

FPGA		Tareas							
Ancho	Alto	Numero	Alto (máx)	Alto (min.)	Ancho (máx.)	Ancho (min.)	Duración máxima	Tiempo final	Ocupación media
10	10	20	8	3	7	2	8	30	38,18 %
10	10	20	5	2	4	1	8	30	21,72%
10	10	20	5	2	4	1	5	30	14,03 %
10	10	15	5	2	4	1	5	30	12,10 %
10	10	15	5	2	4	1	5	20	14,42 %
10	10	15	5	2	4	1	5	15	18,55 %
10	10	15	5	2	4	1	10	15	20,04 %
10	10	15	5	2	4	1	20	15	32,77 %
10	10	15	10	4	4	1	20	15	49,83 %
10	10	15	10	8	10	8	20	15	81,18 %
10	10	25	5	2	4	1	20	30	30,37 %
10	10	25	5	2	4	1	5	30	11,24 %
10	10	30	5	2	4	1	5	30	13,76 %
20	20	20	8	3	7	2	8	30	18,65%
20	20	20	5	2	4	1	8	30	3,98%
20	20	25	5	2	4	1	20	30	8,81 %
20	20	30	5	2	4	1	5	30	5,38 %
20	20	30	15	13	2	1	5	30	13,02 %
20	20	15	5	2	4	1	5	30	2,87 %
20	20	25	5	2	4	1	5	30	3,47 %
20	20	25	20	15	20	15	5	25	56,79 %
15	20	20	8	3	7	2	8	30	19,63 %
15	20	20	5	2	4	1	8	30	7,13 %
10	20	20	8	3	7	2	8	30	27,37 %
10	20	20	5	2	4	1	8	30	12,97 %
10	20	20	10	9	10	9	8	30	54,47 %
10	20	20	10	9	5	4	8	30	44,17 %
30	30	20	10	9	5	4	8	30	11,90 %
30	30	20	15	14	15	14	10	30	42,10 %
30	30	30	15	14	15	14	10	30	52,55 %
30	30	40	15	14	15	14	10	40	54,37 %

FPGA		Tareas							
Ancho	Alto	Numero	Alto (máx)	Alto (min.)	Ancho (máx.)	Ancho (min.)	Duración máxima	Tiempo final	Ocupación media
30	30	35	20	10	20	10	10	40	35,55 %
30	30	35	20	10	20	10	10	40	37,59 %
30	30	70	30	15	30	15	15	70	52,26 %
30	30	50	30	1	30	1	15	60	43,03 %
20	40	50	30	1	20	1	15	60	41,03 %
20	40	20	20	10	20	10	10	40	35,04 %
20	60	20	20	10	20	10	10	40	30,88 %
60	60	100	50	10	50	10	20	110	41,31 %
60	60	100	60	10	60	10	20	110	43,47 %

Bibliografía

1974, *"Worst-case performance bounds for simple one-dimensional packing algorithms"* en SIAM J. Comput., 3, pp. 299-325

Soundara Lakshmi S., *Studies on Maximum Empty Rectangle Problem.* (1986).

B. Chazelle, R. L. Drysdale, and D. T. Lee. Computing the largest empty rectangle. *SIAM Journal on Computing*, 15(1):300-315, February 1986.

Bruce Eckel, President. Thinking in C++ Volume 1 & 2, Second Edition (2000).

Manual de Borland C++

www.cppb.allanpetersen.com/opengl.htm

"Opportunities for Operating Systems Research in Reconfigurable Computing", O. Diessel, G. Wigley, Tech. report ACRC-99-018, 1999.

"Designing Dynamically Recongurable Systems: A High Level Approach", P. Merino, J. C. Lopez, M. J., Proceedings de DCIS'98.

"Fast Template Placement for Reconfigurable Computing Systems", K. Bazargan, R. Kastner, M. Sarrafzadeh, 2000

"On Dynamic Task Scheduling for FPGA-Based Systems", O. Diessel, H. ElGindy, Int. Journal of Foundations of Computer Science, Vol. 12, nº 5, 2001.

"Configuration Relocation and Defragmentation for FPGAs", K. Compton, S. Hauck, IEEE Transactions on VLSI Systems, 2002.

"Non-preemptive Multitasking on FPGAs: Task Placement and Footprint Transform", H. Walder, M. Platzner, ERSA'02

Consultas

E.G. Coffman, Jr., M.R. Garey, and D.S. Johnson. Approximation Algorithms for Bin-Packing -- An Updated Survey. In *Algorithm Design for Computer System Design,* ed. by Ausiello, Lucertini, and Serafini. Springer-Verlag, 1984.

David S. Johnson. Fast Algorithms for Bin Packing. *Journal of Computer and System Sciences* 8, 272-314 (1974).

ANEXOS:

1 . Formato de los ficheros de configuración.
2 . Código fuente.
3 . Manual de usuario

ANEXO 1

FORMATO DE LOS FICHEROS

El ***fichero de tareas*** se guarda con extensión ***.tsk***. El formato del fichero es una lista de tareas con los siguientes atributos:

Ancho_tarea1 Alto_tarea1 Tiempo de Llegada_tarea1 Duración_tarea1 Estado_tarea1
Ancho_tarea2 Alto_tarea2 Tiempo de Llegada_tarea2 Duración_tarea2 Estado_tarea2
....
....
Ancho_tareaN Alto_tareaN Tiempo de Llegada_tareaN Duración_tareaN Estado_tareaN

El ***fichero FPGA*** guarda la información referente a la fpga tiene extensión ***.fpg.*** El formato del fichero es el siguiente:

Alto_FPGA Ancho_FPGA Nombre_FPGA

El ***fichero de configuración*** almacena todos los datos correspondientes a una simulación. Esto quiere decir que guarda toda la información de la FPGA, de las tareas y del algoritmo utilizado. El fichero tiene extensión ***.cfg*** y es de la siguiente manera:

FPGA
Alto_FPGA Ancho_FPGA Nombre_FPGA

TAREAS
Ancho_tarea1 Alto_tarea1 Tiempo de Llegada_tarea1 Duración_tarea1 Estado_tarea1
Ancho_tarea2 Alto_tarea2 Tiempo de Llegada_tarea2 Duración_tarea2 Estado_tarea2
....
....
Ancho_tareaN Alto_tareaN Tiempo de Llegada_tareaN Duración_tareaN Estado_tareaN

ALGORITMO
Nombre_Algoritmo_Utilizado

END

ANEXO 2

CÓDIGO FUENTE

Fichero Acerca.cpp

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Acerca.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TAcercade *Acercade;
//---------------------------------------------------------------------------
__fastcall TAcercade::TAcercade(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
```

Fichero BinPackingMER.cpp

```
#include "Utils.h"
#include "Point2D.h"
#include "BinPackingMER.h"
#include <strstream>
#include <vector>
using namespace std;
#include <stdio.h>
//---------------------------------------------------------------------
/** Destructor de la clase */
BinPackingMER::~BinPackingMER(){
      if (DEBUG) ECHO("Entrando al Destructor de BinPackingMer");
      if (this->fpga != NULL) delete this->fpga;
      if (this->log != NULL) delete this->log;
}
//---------------------------------------------------------------------
// METODOS PRIVADOS
//-----------------------------------------L---------------------
/** Metodo de inicializacion de datos privados de la clase. Limpia las listas
de tareas y
la lista de MER */
int BinPackingMER::_Init(){
      if (DEBUG) ECHO("Entrando al metodo _Init de BinPackingMer");
  // Sembramos la semilla aleatoria
  srand(time(0));
      // Inicializamos nuestras variables
      this->fpga = new FPGA(this->W,this->H);
      this->log = new Log("Historico.txt");
      // Copiamos la lista a la lista de espera la lista inicial.
      this->listaEspera = listaInicial;
      // Limpiamos la lista de tareas activas
      this->listaActual.clear();
      // Limpiamos la lista de MERs
      this->listaMER.clear();
```

```
  this->ocupacionInst = 0.0f;
  this->ocupacionMediaAc = 0.0f;
  this->numEventos = 0;
  this->comentario = "BinPacking MER";
      this->log->write("Altura: %i, Anchura:%i, Nombre:%i.\n",this->H,this-
>W,this->nombreFPGA.c_str());
      return 0;
}
//-----------------------------------------------------------------
/** Metodo encargado de vaciar la lista de tareas activas cuando estas salgan
de la fpga */
int BinPackingMER::acabaTarea(){
      int retVal = 0; // indica el numero de tareas que han acabado en este
instante de tiempo
      if (DEBUG) ECHO("Entrando al metodo acabaTarea() de BinPackingMer");
      // Si la lista esta vacia, salimos del metodo
      if (this->listaActual.size() == 0){
            if (DEBUG) ECHO("acabaTarea -> Lista de tareas activa vacia");
            return 0;
      }
      // Nos recorremos toda la lista de tareas activas
  this->it = this->listaActual.begin();
      while (this->it != this->listaActual.end()){
            SituacionTarea aux = *(this->it);
            Task tAct = aux.tarea;
            // La tarea ha acabado
            if ((tAct.getTStart() + tAct.getLength()) == this->tActual) {
      list<SituacionTarea>::iterator itAux;
                  // Quitamos la tarea de la lista activa
                  if (DEBUG) cout << "Quitamos la tarea colocada en " << aux.x
<< "," << aux.y << ")" << endl;
                  log->write("Tiempo: %i. Sale la tarea colocada en %i,%i, con
altura:%i y anchura:%i.\n",this-
>tActual,aux.x,aux.y,aux.tarea.getH(),aux.tarea.getW());
      itAux = it;
      itAux++;
                  this->listaActual.erase(it);
      it = itAux;
                  retVal++;
            }
    else it++;
      }
  this->numEventos++;
  this->ocupacionInst = this->calcularOcupInst();
  // TODO: Calcular la ocupacion media bien...
  this->ocupacionMediaAc += this->ocupacionInst;
      return retVal;
}
//-----------------------------------------------------------------
/* Devuelve el numero de tareas que empiezan en este instante de tiempo
determinado.
-1 en caso de error */
int BinPackingMER::empiezaTarea(){
      if (DEBUG) ECHO("Entrando al metodo empiezaTarea() de BinPackingMer");
      // Recuperamos la primera tarea en la lista de espera
      Task tFirst = *(this->listaEspera.begin());
      // Empieza alguna tarea en el tiempo actual?
      if (this->tActual == tFirst.getTStart()){
            // Recalculamos la lista de rectangulos vacios
            if (this->calcularMER()<0){
                  if (DEBUG) this->log->write("Error al calcular el MER");
                  return -1;
```

```
            }
            // TODO: Comprobar que la tarea quepa en el rectangulo vacio !!
            list<CRectangle>::iterator itMER;
    // DEPURACION
    for (itMER = this->listaMER.begin(); itMER != this-
>listaMER.end();itMER++){
            CRectangle MER = *(itMER);
      this->log->write("Rectangulo MER : (%i,%i) -
(%i,%i)",MER.x1,MER.y1,MER.x2,MER.y2);
    }

    itMER = this->listaMER.begin();
            CRectangle MER = *(itMER);
            CRectangle test(MER.x1,MER.y1,MER.x1 + tFirst.getW(),MER.y1 +
tFirst.getH());
            bool cabe = false; // Flag para indicar si cabe o no una tarea
            while ((!cabe) && (itMER != this->listaMER.end()) &&
((*itMER).area() >= test.area())){
                  // Si el rectangulo MER contiene a la tarea
                  if (MER.contiene(test)) {
                        cabe = true;
                        break;
                  }
                  itMER++;
                  MER = *(itMER);
                  test = CRectangle(MER.x1,MER.y1,MER.x1 +
tFirst.getW(),MER.y1 + tFirst.getH());
            }
            // La tarea no cabe en ningun rectangulo vacio
            if (!cabe) return -1;
            this->listaEspera.pop_front(); // Eliminamos la primera tarea de
la lista de tareas
    // Introducimos la tarea en la esquina inferior izquierda del primer
rectagulo vacio
            SituacionTarea tareaEntrante;
    // Generamos el color de la tarea
    // Color aleatorio de las tareas
    tareaEntrante.r = rand() % 255;
    tareaEntrante.g = rand() % 255;
    tareaEntrante.b = rand() % 255;
            tareaEntrante.tarea = tFirst;
            tareaEntrante.x = MER.x1;
            tareaEntrante.y = MER.y1;
            this->log->write("Tiempo %i: Entra tarea en (%i,%i), con
anchura=%i,altura=%i",this-
>tActual,tareaEntrante.x,tareaEntrante.y,tFirst.getW(),tFirst.getH());
            this->listaActual.push_back(tareaEntrante); // La añadimos a la
lista de tareas
    this->numEventos++;
    this->ocupacionInst = this->calcularOcupInst();
    this->ocupacionMediaAc += this->ocupacionInst;
            return 1;
      }
      else return 0;
}
//-----------------------------------------------------------------
/** Inserta de forma ordenada un rectángulo en la lista de Rectángulos
Máximos.
Si está contenido en algún rectángulo de la lista no lo mete.*/
int BinPackingMER::insertMER(CRectangle r,list <CRectangle> &listaMER){
      listaMER.reverse();  // Ordenamos ascendentemente
      list <CRectangle>::iterator itMER = listaMER.begin();
```

```
	// Si el rectangulo es contenido por alguno existente, no lo metemos en
la lista
	while (itMER != listaMER.end()){
		if((*itMER).contiene(r))
			return 1;
		itMER++;
	}
	// Miramos si el rectangulo contiene a algun rectangulo, y si es asi lo
eliminamos de la
	// lista
	itMER = listaMER.begin();
	while (itMER != listaMER.end()){
		if (r.contiene((*itMER))) {
			listaMER.erase(itMER);
			break;
		}
		itMER++;
	}
	listaMER.push_back(r);
	listaMER.sort();
	listaMER.reverse(); // Ordenamos ascendentemente
	if (DEBUG){
		ECHO("======= Nueva Lista MER ======");
		for (itMER = listaMER.begin() ; itMER != listaMER.end(); itMER++){
			CRectangle curr = (*itMER);
			cout << "Rectangulo = (" << curr.x1 << "," << curr.y1 << ")
- (" << curr.x2 << "," << curr.y2 << ")" << endl;
		}

	}
	return 0;
}
//---------------------------------------------------------------
/** Metodo de calculo del conjunto de rectangulo maximales vacios.
Devuelve el numero de MER que se han formado. -1 en caso de error.
Pasos a realizar:
1. Si la FPGA esta vacia, el unico rectangulo es la propia FPGA
2. Si no esta vacia:
	2.1. Calcular la lista de puntos de interseccion de las tareas
	con la pared de la FPGA.
	2.2. Conseguir rectangulo vacios en la fpga. Si ya esta en la lista MER
o esta contenido por alguno de la lista
	no se introduce en la lista. Si no, lo metemos en la lista
*/
int BinPackingMER::calcularMER(){
	if (DEBUG) ECHO("Entrando al metodo calcularMER() de BinPackingMer");
	this->listaMER.clear(); // Limpiamos la lista anterior de rectangulos
vacios
	// Inicializa una matriz de booleanos indicando en qué posiciones hay
tarea.
	int longitud = this->fpga->getH() * this->fpga->getW();
	int filas = this->fpga->getH();
	int cols = this->fpga->getW();
	bool *MFPGA = new bool[this->fpga->getH() * this->fpga->getW()];
	for (int i=0;i<longitud;i++)
			MFPGA[i]  = false;
	for(this->it = this->listaActual.begin();this->it!=this-
>listaActual.end();this->it++){
		SituacionTarea sit= *it;
		for (int j=sit.y;j<(sit.y + sit.tarea.getH());j++)
			for (int i=sit.x;i<(sit.x+sit.tarea.getW());i++){
				if (DEBUG)
```

```
                              ECHO_VAR("Poniendo a true la casilla = ",j *
cols + i);
                        MFPGA[j * cols + i] = true;
                  }
      }
      if(this->listaActual.size() == 0) {
            if (DEBUG) ECHO("calcularMER : La fpga esta vacia");
            CRectangle r(0,0,this->fpga->getW(),this->fpga->getH());
            this->listaMER.push_back(r);
            if (DEBUG)
                  ECHO_VAR("Saliendo al metodo calcularMER() de BinPackingMer
con valor de retorno = ",this->listaMER.size());
            return this->listaMER.size();
      }
      // Lista de puntos de corte de las tareas con la pared izquierda de la
FPGA
      vector<int> puntosCorte;
      // Pared izquierda de la fpga
      Point2D A(0,0);
      Point2D B(0,this->fpga->getH());
      Point2D in; // Punto de interseccion con la pared de la fpga
      puntosCorte.push_back(0); // Metemos el punto inferior de la FPGA
      /* Nos recorremos toda la lista actual para hallar los puntos de
interseccion de las rectas horizontales
      de las tareas con la pared izquierda de la fpga */
      for (this->it = this->listaActual.begin();this->it != this-
>listaActual.end();it++){
            SituacionTarea st = *(this->it); // Conseguimos la situacion de la
tarea
            // Formamos las rectas de la tarea
            Point2D a(st.x,st.y); // Recta inferior
            Point2D b(st.x + st.tarea.getW(),st.y);
            Point2D c(st.x ,st.y + st.tarea.getH()); // Recta superior
            Point2D d(st.x + st.tarea.getW(),st.y + st.tarea.getH());
            if (interseccion(A,B,a,b,in) < 0) { // Interseccion de la recta
inferior con la pared de la fpga
                  if (DEBUG) ECHO("Parece que no hay interseccion con la
fpga");
                  return -1;
            }
            if ((int) in.getY() != 0) puntosCorte.push_back((int) in.getY());
            if (interseccion(A,B,c,d,in) < 0) { // Interseccion de la recta
superior con la pared de la fpga
                  if (DEBUG) ECHO("Parece que no hay interseccion con la
fpga");
                  return -1;
            }
            if ((int) in.getY() != 0) puntosCorte.push_back((int)in.getY());
      }
      // Añadimos el punto superior de la pared de la FPGA
      if (puntosCorte[puntosCorte.size() - 1] != this->fpga->getH())
puntosCorte.push_back(this->fpga->getH());
      // Para cada punto de corte...
      for (int i=0;i<puntosCorte.size()-1;i++){
            Point2D a,b,c,d;
            // Formamos los rectangulos vacios que haya en la fpga
            for (int j=i+1;j<puntosCorte.size();j++){
                  int k=0;
                  int vertice = puntosCorte[i];
                  int vertice2 = puntosCorte[j];
                  while (k<this->fpga->getW()){
        // Mientras hay tarea
```

```
                        while ((hayTarea(MFPGA,vertice,vertice2,k) == 0) && (k
< this->fpga->getW()))
                              k++;
                        a = Point2D(k,vertice);
                        c = Point2D(k,vertice2);
                        while ((hayTarea(MFPGA,vertice,vertice2,k) == 1) && (k
< this->fpga->getW()))
                              k++;
                        b=Point2D(k,vertice);
                        d=Point2D(k,vertice2);
                        CRectangle rect(a.getX(),a.getY(),d.getX(),d.getY());
                        this->insertMER(rect,listaMER);
                  }
            }
      }
      if (DEBUG) ECHO("Creación lista MER finalizada");
      if (MFPGA != NULL) delete MFPGA;
      return 0;
}
//-----------------------------------------------------------------
/* Metodo que nos indica si hay o no hay tarea en el rango de filas indicado
por i,j en la columna k.
*/
int BinPackingMER::hayTarea(bool *MFPGA,int i,int j,int k){
  if (MFPGA == NULL) return -1;
      int filas = this->fpga->getH();
      int cols = this->fpga->getW();
  bool retVal = false;
  int index = i;
  while ((index < j) && (!retVal)) {
    retVal = MFPGA[index * cols + k];
    index++;
  }
  if (retVal) return 0;
  else return 1;
}
//-----------------------------------------------------------------
// Devuelve la ocupacion instantanea de la fpga del apgoritmo. -1 en caso de
error
float BinPackingMER::calcularOcupInst(){
  if (this->fpga == NULL) return -1.0f;
  float retVal = 0.0f;
  int sumAreas = 0; // Suma de areas de la tareas activas
  int areaFpga = this->fpga->getH() * this->fpga->getW();
  // Para todas las tareas activas
      for (this->it = this->listaActual.begin();this->it != this-
>listaActual.end();it++){
    SituacionTarea sit= *it;
    Task t = sit.tarea;
    int areaI =  t.getH() * t.getW();
    sumAreas += areaI;
  }
  retVal = (sumAreas == 0) ? 0.0f :  (float) sumAreas /(float) areaFpga;
  return retVal;
}
//-----------------------------------------------------------------
// METODOS DE LA INTERFAZ
//-----------------------------------------------------------------
void BinPackingMER::Init(list<Task> listaInicial,int H,int W,string
nombreFPGA){
      if (DEBUG) ECHO("Entrando al metodo Init de BinPackingMer");
      this->tActual = 0;
```

```
      this->listaInicial = listaInicial;
      this->H = H;
      this->W = W;
      this->nombreFPGA = nombreFPGA;
      this->_Init();
}
//---------------------------------------------------------------------
int BinPackingMER::step(){
      if (DEBUG) ECHO("Entrando al metodo step() de BinPackingMer");
      // Da un paso del algoritmo.
      /*
      Casos que se pueden dar:
      1. No sale ni entra una nueva tarea.
      2. Entra pero no sale una tarea.
      3. Sale pero no entra una tarea.
      4. Sale y entra una nueva tarea.
      */
      // Si las dos listas, de espera y activa estan vacias, hemos acabado el
algoritmo
      if ((this->listaEspera.size() == 0) && (this->listaActual.size() == 0)){
            if (DEBUG) ECHO("El algoritmo ha acabado de procesar todas las
tareas");
            return 2;
      }
  bool hay_evento = false;
  int retVal1 = this->acabaTarea();
      if ( retVal1 < 0){
            if (DEBUG) WARNING("Error al llamar al metodo acabaTarea");
            return -1;
      }
  int retVal2 = this->empiezaTarea();
      if (retVal2 < 0){
            // No podemos meter la tarea en ningun restangulo vacio
            if (DEBUG) WARNING("La tarea no cabe en los rectangulos
vacios..");
            list<Task>::iterator itTask;
            // Aumentamos en 1 el tiempo de las tareas en espera
            for (itTask = this->listaEspera.begin(); itTask != this-
>listaEspera.end();itTask++)
                  (*itTask).setTStart((*itTask).getTStart() + 1);
      }
  // Hay evento si ha entrado o ha salido alguna tarea
  hay_evento = (retVal1 > 0) || ( retVal2 >0);
      // Incrementamos en uno el tiempo del algoritmo
      this->tActual++;
  if (hay_evento) return 1; // Devolvemos 1 si se ha producido algun evento
  else return 0;
}
//---------------------------------------------------------------------
int BinPackingMER::next_event(){
      if (DEBUG) ECHO("Entrando al metodo next_event() de BinPackingMer");
      // Si las dos listas, de espera y activa estan vacias, hemos acabado el
algoritmo
      if ((this->listaEspera.size() == 0) && (this->listaActual.size() == 0)){
            if (DEBUG) ECHO("El algoritmo ha acabado de procesar todas las
tareas");
            return 2;
      }
  int retVal;
  while((retVal = this->step()) == 0)
    continue;
  if (retVal == 1) // Ya se ha producido un evento
```

```
    return 0;
  if (retVal == 2) // Se ha acabado el algoritmo
    return 2;
  if (retVal == -1) // Se ha producido un error
    return -1;
}
//---------------------------------------------------------------------
int BinPackingMER::run(){
      if (DEBUG) ECHO("Entrando al metodo run() de BinPackingMer");
      while (this->step() != 1);
      return 0;
}
//---------------------------------------------------------------------
int BinPackingMER::reset(){
      if (DEBUG) ECHO("Entrando al metodo reset() de BinPackingMer");
      // Ponemos a cero el contador de tiempo.
      this->log->write("Reseteamos el algoritmo...");
      this->tActual = 0;
      this->_Init(); // Reinicializamos las variables de la clase
      return 0;
}
//---------------------------------------------------------------------
string BinPackingMER::toString(){
      if (DEBUG) ECHO("Entrando al metodo toString() de BinPackingMer");
      ostrstream s;
      s << "Clase BinPackingMER:" << endl;
      s << "Tiempo actual = " << this->tActual << endl;
      if (this->fpga != NULL) s << "Fpga = " << this->fpga->getH() << "," <<
this->fpga->getW() << endl;
      s << "Longitud de la lista actual = " << this->listaActual.size() <<
endl;
      s << "Longitud de la lista espera = " << this->listaEspera.size() <<
endl;
      s << "Longitud de la lista inicial = " << this->listaInicial.size() <<
endl;
      s << ends;
      string retVal;
      retVal = s.str();
      // Cosas que hay que hacer para liberar el stream
      s.seekp(-1, ios::cur);
      s.rdbuf()->freeze(0);
      return retVal;
}
//---------------------------------------------------------------------
float BinPackingMER::getOcupacionInst(){
  return round2Dec(this->ocupacionInst * 100.0f);
}
//---------------------------------------------------------------------
float BinPackingMER::getOcupacionMedia(){
  // Prevenimos division entre cero
  if (this->numEventos == 0.0f) return 0.0f;
  float retVal = round2Dec((this->ocupacionMediaAc/this->numEventos) *
100.0f);
  return retVal;
}
```

Fichero Crectangle.cpp

```
#include "CRectangle.h"
#include "Utils.h"
//---------------------------------------------------------------
/** Constructor de la clase */
CRectangle::CRectangle(int x1,int y1,int x2,int y2){
      this->x1 = x1;
      this->x2 = x2;
      this->y1 = y1;
      this->y2 = y2;
}
//---------------------------------------------------------------
CRectangle::CRectangle(){
      this->x1 = 0;
      this->x2 = 0;
      this->y1 = 0;
      this->y2 = 0;
}
//---------------------------------------------------------------
CRectangle::CRectangle(const CRectangle& r){
      this->x1 = r.x1;
      this->x2 = r.x2;
      this->y1 = r.y1;
      this->y2 = r.y2;
}
//---------------------------------------------------------------
CRectangle::~CRectangle(){
}
//---------------------------------------------------------------
/** Devuelve el area de un rectangulo */
int CRectangle::area(){
      // if (DEBUG) ECHO("Entrando en el metodo area() de CRectangle");
      if (isEmpty()) return 0;
      else return ((x2 - x1) * (y2 - y1));
}
//---------------------------------------------------------------
/** Indica si un rectangulo está o no vacío */
bool CRectangle::isEmpty(){
      // if (DEBUG) ECHO("Entrando en el metodo isEmpty() de CRectangle");
      if ((x2 == x1) && (y2 == y1)) return true;
      else return false;
}
//---------------------------------------------------------------
/** Funcion que indica si un rectangulo contiene a otro */
bool CRectangle::contiene(const CRectangle &r){
      if ((x1 <= r.x1) && (y1 <= r.y1) && (x2 >= r.x2) &&  (y2 >= r.y2))
return true;
      else return false;
}

//---------------------------------------------------------------------------
```

Fichero DatosFPGA.cpp

```
#include <vcl.h>
#pragma hdrstop

#include "DatosFPGA.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
```

```
#pragma resource "*.dfm"
TFormFPGA *FormFPGA;
//---------------------------------------------------------------------------
__fastcall TFormFPGA::TFormFPGA(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TFormFPGA::GenerarClick(TObject *Sender)
{
  this->ModalResult = mrOk;
}
//---------------------------------------------------------------------------
```

Fichero DatosLista.cpp

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "DatosLista.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TListaTareas *ListaTareas;
//---------------------------------------------------------------------------
__fastcall TListaTareas::TListaTareas(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TListaTareas::GenerarClick(TObject *Sender)
{
AnsiString aux;
datosMal=false;
//Comprobamos todos los datos antes de salir del formulario
      try {
        aux = this->numtareasF->Text;
        numtareas = StrToInt(aux);
        if (numtareas<=0) datosMal=true;
    }
      catch (Exception &error){
         datosMal=true;
         return;
    }
      try {
        aux = this->altomaxF->Text;
        altomax = StrToInt(aux);
        if (altomax<=0) datosMal=true;
    }
      catch (Exception &error){
         datosMal=true;
         return;
    }
      try {
        aux = this->altominF->Text;
        altomin = StrToInt(aux);
        if ((altomin<=0)||(altomin>altomax)) datosMal=true;
```

```
    }

      catch (Exception &error){
         datosMal=true;
         return;
    }
      try {
        aux = this->anchomaxF->Text;
        anchomax = StrToInt(aux);
        if (anchomax<=0) datosMal=true;
    }
      catch (Exception &error){
         datosMal=true;
         return;
    }
      try {
        aux = this->anchominF->Text;
        anchomin = StrToInt(aux);
        if ((anchomin<=0)||(anchomin>anchomax)) datosMal=true;
    }
      catch (Exception &error){
         datosMal=true;
         return;
    }
      try {
        aux = this->durmaxF->Text;
        durmax = StrToInt(aux);
        if (durmax<=0) datosMal=true;
    }
      catch (Exception &error){
         datosMal=true;
         return;
    }
}
//---------------------------------------------------------------------------

void __fastcall TListaTareas::CancelarClick(TObject *Sender)
{
  this->ModalResult = mrCancel;
}
//---------------------------------------------------------------------------
```

Fichero FPGA.cpp

```
#include "FPGA.h"
#include "Utils.h"
#include <strstream>
using namespace std;
//-----------------------------------------------------
FPGA::FPGA(int W,int H){
      this->W = W;
      this->H = H;
}
//-----------------------------------------------------
FPGA::FPGA(int W,int H,string n){
      this->W = W;
      this->H = H;
  this->name = n;
}
//-----------------------------------------------------
FPGA::~FPGA(){
```

```
}
//-------------------------------------------------------
int FPGA::getH(){ return this->H ;}
//-------------------------------------------------------
int FPGA::getW(){ return this->W ; }
//-------------------------------------------------------
/** Metodo para volcar los datos a una cadena */
string FPGA::toString(){
      ostrstream s;

      s << "Clase FPGA:" << endl;
      s << "Ancho = " << this->W << endl;
      s << "Alto = " << this->H << endl;
      s << ends;
      string retVal;
      retVal = s.str();
      // Cosas que hay que hacer para liberar el stream
      s.seekp(-1, ios::cur);
      s.rdbuf()->freeze(0);
      return retVal;
}
```

Fichero Log.cpp

```
//---------------------------------------------------------------------------
#include "Log.h"
#include <stdarg.h>
#include <time.h>
#include <errno.h>

bool Log::Init(char *path){
        // Abrimos el fichero para añadir al final
        if ((this->logfile = fopen(path, "a")) == NULL)
        {
          fprintf(stderr, "Cannot open input file for writing.\n");
          return false;
        }
        this->path = path;
        return true;
}
//---------------------------------------------------------------------------
bool Log::Shutdown(){
        fflush(logfile);
        if (fclose(this->logfile) == EOF)
        {
                fprintf(stderr, "Cannot close file for writing.\n");
                return false;
        }
        // TODO : no hay que liberar el path??
        return true;
}
//---------------------------------------------------------------------------
bool Log::write(char *text,...){
      if (this->_debug){
            va_list args;
            time_t t;
            struct tm tm;
            char buf[256];
            int errno_save = errno;
```

```
            va_start(args, text);
            time(&t);
            tm = *localtime(&t);
            strftime(buf, sizeof(buf)-1, "[%b %d %H:%M:%S %Y] ", &tm);
            fputs(buf,logfile);
            vfprintf(logfile, text, args);
            fputc('\n', logfile);
            //this->Shutdown();
            //this->Init(this->path);
      }
      return true;
}
//---------------------------------------------------------------------------
bool Log::write(string text){
      if (this->_debug){
            time_t t;
            struct tm tm;
            char buf[256];
            int errno_save = errno;
            time(&t);
            tm = *localtime(&t);
            strftime(buf, sizeof(buf)-1, "[%b %d %H:%M:%S %Y] ", &tm);
            fputs(buf,logfile);
            fprintf(logfile, text.c_str());
            fputc('\n', logfile);
            //this->Shutdown();
            //this->Init(this->path);
      }
      return true;
}
//---------------------------------------------------------------------------
void Log::setDebug(bool debug){
      this->_debug = debug;
}
```

Fichero Ppal.cpp

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop
#include<iostream>
#include<fstream>
#include<iomanip>
using namespace std;
#include "Ppal.h"
#include "DatosLista.h"
#include "VerHistorico.h"
#include "DatosFPGA.h"
#include "Acerca.h"
#include "Utils.h"
#include "BinPackingMER.h"
#include <math.h>
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma link "TOpenGLAPPanel"
#pragma resource "*.dfm"
TMain *Main;
//---------------------------------------------------------------------------
__fastcall TMain::TMain(TComponent* Owner)
```

```
        : TForm(Owner)
{
  // Creamos las cabeceras de la tabla
  this->TablaTareas->Cells[0][0] = "Orden";
  this->TablaTareas->Cells[1][0] = "Ancho";
  this->TablaTareas->Cells[2][0] = "Alto";
  this->TablaTareas->Cells[3][0] = "Duracion";
  this->TablaTareas->Cells[4][0] = "T. Llegada";
  this->TablaTareas->Cells[5][0] = "Estado";
  this->TablaTareas->Cells[6][0] = "Color";
  this->updateDatosFPGA();
  this->fpga = new FPGA();
  this->log = new Log("TMain.log");
  this->updateStatusBar();
}
//---------------------------------------------------------------------------
void __fastcall TMain::Salir1Click(TObject *Sender)
{
  // TODO: Poder guardar los cambios
  if (this->listaTareas.size() != 0) {
    //ShowMessage("La lista de Tareas ha sido modificada ¿Desea Guardar los
cambios?");
  }
  if (this->fpga != NULL){
    //ShowMessage("La FPGA ha sido modificada ¿Desea Guardar los cambios?");
  }
  Application->Terminate();
}
//---------------------------------------------------------------------------

void __fastcall TMain::NuevaListaTareas1Click(TObject *Sender)
{

  AnsiString datos;
  if (ListaTareas->ShowModal() == mrOk){
    int minH,minW,maxH,maxW;
    int numTareas;
    int durMax;
    int tiempoFinal;
    datos = ListaTareas->numtareasF->Text;
    try {
      numTareas = StrToInt(datos);
    }
    catch (Exception &error){
      ShowMessage("El número de tareas tiene que ser un número entero
positivo.");
      return;
    }
    datos = ListaTareas->altomaxF->Text;
    try {
      maxH = StrToInt(datos);
    }
    catch (Exception &error){
      ShowMessage("El alto maximo tiene que ser un número entero positivo.");
      return;
    }
    datos = ListaTareas->altominF->Text;
    try {
      minH = StrToInt(datos);
    }
    catch (Exception &error){
      ShowMessage("El alto minimo tiene que ser un número entero positivo.");
```

```
      return;
    }
    datos = ListaTareas->anchomaxF->Text;
    try {
      maxW = StrToInt(datos);
    }
    catch (Exception &error){
      ShowMessage("El ancho maximo tiene que ser un número entero positivo.");
      return;
    }
    datos = ListaTareas->anchominF->Text;
    try {
      minW = StrToInt(datos);
    }
    catch (Exception &error){
      ShowMessage("El ancho minimo tiene que ser un número entero positivo.");
      return;
    }
    datos = ListaTareas->durmaxF->Text;
    try {
      durMax = StrToInt(datos);
    }
    catch (Exception &error){
      ShowMessage("La duracion maxima tiene que ser un número entero
positivo.");
      return;
    }
    datos = ListaTareas->tiempoFinal->Text;
    try {
      tiempoFinal = StrToInt(datos);
    }
    catch (Exception &error){
      ShowMessage("La duracion maxima tiene que ser un número entero
positivo.");
      return;
    }
    // Los datos son correctos ??
    if(!this->checkDatosFPGA(*fpga,minH,maxH,minW,maxW,numTareas,tiempoFinal))
      return;
    // Creamos la lista de tareas
    createTasks(minW,maxW,minH,maxH,durMax,tiempoFinal,numTareas,*(this-
>fpga),this->listaTareas);
    this->updateTablaTareas();
    if (this->alg != NULL){
        this->alg = new BinPackingMER();
        this->alg->Init(this->listaTareas,this->fpga->getH(),this->fpga-
>getW(),
                    this->fpga->getName());
        this->updateTiempoActual();
    }
  }
}
//---------------------------------------------------------------------------
void __fastcall TMain::AbrirListaTareas1Click(TObject *Sender)
{
  if (AbrirLista->Execute()){
    if(loadTasks(AbrirLista->FileName.c_str(),this->listaTareas)<0){
      ShowMessage("Error al cargar la lista de tareas...");
      return;
    }
    else {
      this->updateTablaTareas();
```

```
      this->updateTablaTareas();
    if (this->alg!=NULL){
        this->alg = new BinPackingMER();
        this->alg->Init(this->listaTareas,this->fpga->getH(),this->fpga-
>getW(),
                    this->fpga->getName());
        this->updateTiempoActual();
    }
      ShowMessage("Lista cargada con exito!");
    }
  }
}
//---------------------------------------------------------------------------
void __fastcall TMain::AbrirFPGA1Click(TObject *Sender)
{
  if (AbrirFPGA->Execute()){
    if (this->fpga == NULL) this->fpga = new FPGA();
    if (loadFPGA(AbrirFPGA->FileName.c_str(),*fpga)<0){
      ShowMessage("Error al cargar la fpga. ");
      return;
    }
    this->updateDatosFPGA();
    this->setPanelFPGA();
  }
}
//---------------------------------------------------------------------------
void __fastcall TMain::GuardarListaTareas1Click(TObject *Sender)
{
  if (GuardarLista->Execute()) {
    if (saveTasks(GuardarLista->FileName.c_str(),this->listaTareas) <0){
      ShowMessage("Error al guardar la lista de tareas ");
      return;
    }
    else ShowMessage("Lista guardada con exito!");
  }
}
//---------------------------------------------------------------------------
void __fastcall TMain::GuardarFPGA1Click(TObject *Sender)
{
  if (GuardarFPGA->Execute()){
    if (saveFPGA(GuardarFPGA->FileName.c_str(),*fpga)<0){
      ShowMessage("Error al guardar fpga a disco");
      return;
    }
    else ShowMessage("Fpga guardada con exito!");
  }
}
//---------------------------------------------------------------------------
void __fastcall TMain::Acercade1Click(TObject *Sender)
{
  Acercade->ShowModal();

}
//---------------------------------------------------------------------------
void __fastcall TMain::Neventos1Click(TObject *Sender)
{
  if (this->alg == NULL) {
    ShowMessage("Debe seleccionar un algoritmo antes.");
    return;
  }
  AnsiString aux;
  bool datosMal = true;
```

```
  int eventos;
  do {
    if (InputQuery("Número de eventos","Introduce el número de eventos",
aux)){
      try {
        eventos = StrToInt(aux);
        if (eventos<=0)
          ShowMessage("El número de eventos tiene que ser un número entero
positivo");
        else
          datosMal = false;
      }
      catch (Exception &error){
        ShowMessage("El número de eventos tiene que ser un número entero
positivo");
      }
    }
    else
      return; // Hemos pulsado el boton de cancelar
  } while (datosMal);
  // Llamamos n-veces al metodo de eventos
  for (int i=0;i<eventos;i++){
    int retVal = this->alg->next_event();
    if (retVal == 2) {
      ShowMessage("La ejecucion del algoritmo ha finalizado");
      return;
    }
    if (retVal == -1) {
      ShowMessage("La ejecucion del algoritmo ha producido error");
      return;
    }
  }
  this->updateStatusBar();
  this->updateTiempoActual();
  this->updateTablaTareas();
  this->OpenGLAPPanel1Paint(Sender);
}
//---------------------------------------------------------------------------
void __fastcall TMain::NPasos1Click(TObject *Sender)
{
  AnsiString aux;
  bool datosMal = true;
  int pasos;
  do {
    if (InputQuery("Número de pasos","Introduce el número de pasos", aux)){
      try{
        pasos = StrToInt(aux);
        if (pasos<=0) {
          ShowMessage("El número de pasos tiene que ser un número entero
positivo");
        }
        else datosMal=false;
      }
      catch (Exception &error){
        ShowMessage("El número de pasos tiene que ser un número entero
positivo");
      }
    }
    else
      return; // Hemos pulsado el boton de cancelar
  } while (datosMal);
```

```
  // Llamamos n-veces al metodo de pasos
  for (int i=0;i<pasos;i++){
    int retVal = this->alg->step();
    if (retVal == 2) {
      ShowMessage("La ejecucion del algoritmo ha finalizado");
      return;
    }
    if (retVal == -1) {
      ShowMessage("La ejecucion del algoritmo ha producido error");
      return;
    }
  }
  this->updateStatusBar();
  this->updateTablaTareas();
  this->updateTiempoActual();
  this->OpenGLAPPanel1Paint(Sender);
}
//---------------------------------------------------------------------------
// Metodo de refresco de la tabla de tareas que hay en pantalla
void __fastcall TMain::updateTablaTareas(){
  this->TablaTareas->RowCount = 2;
  list<Task>::iterator it;
  int i = 0;
  for (it=this->listaTareas.begin();it != this->listaTareas.end();it++,i++){
    Task t = (*it);
    this->TablaTareas->Cells[0][i+1] = i; // Numero de tarea
    this->TablaTareas->Cells[1][i+1] = t.getW(); // Ancho
    this->TablaTareas->Cells[2][i+1] = t.getH(); // Alto
    this->TablaTareas->Cells[3][i+1] = t.getLength(); // Duracion
    this->TablaTareas->Cells[4][i+1] = t.getTStart(); // Tiempo de llegada
    // Comprobacion del estado de las tareas
    string cad;
    int estado = this->getEstado(t);
    switch(estado){
      case(TAREA_LIB):
        cad = "Libre";
        break;
      case(TAREA_ESPERA):
        cad ="Espera";
        break;
      case(TAREA_EJEC):
        cad = "Ejecucion";
        break;
      default:
        cad = "Error";
    }
    this->TablaTareas->Cells[5][i+1] = cad.c_str(); // Estado
    if (estado == TAREA_EJEC){
      this->TablaTareas->Font->Color = 0x00FF0000;
      this->TablaTareas->Cells[6][i+1] = "COLOR"; // color
      this->TablaTareas->Font->Color = clBlack;
    }
    else
      this->TablaTareas->Cells[6][i+1] = "";
    this->TablaTareas->RowCount++;
  }
}
//---------------------------------------------------------------------------
void __fastcall TMain::Seleccionar1Click(TObject *Sender)
{
    if (this->fpga == NULL) {
      ShowMessage("Debe inicializar la FPGA del programa");
```

```
      return;
    }
    this->alg = new BinPackingMER();
    this->alg->Init(this->listaTareas,this->fpga->getH(),this->fpga->getW(),
                    this->fpga->getName());
    this->updateTiempoActual();

    this->updateStatusBar();
}
//---------------------------------------------------------------------------
void __fastcall TMain::Paso1Click(TObject *Sender)
{
  if (this->alg==NULL){
    ShowMessage("Debes seleccionar primero un algoritmo");
    return;
  }
  int retVal = this->alg->step();
  if (retVal == 2) // Acabado el algoritmo
    ShowMessage("Acabado el algoritmo");
  if (retVal == -1) // Error al ejecutar un paso
    ShowMessage("Error al dar un paso en el algortimo");
  if (retVal == 1){
    this->updateTablaTareas();
    this->OpenGLAPPanel1Paint(Sender);
  }
  this->updateTiempoActual();
  this->updateStatusBar();
  return;
}
//---------------------------------------------------------------------------
// Metodo de refresco de la etiqueta con el tiempo actual
void __fastcall TMain::updateTiempoActual()
{
  if (this->alg != NULL)
    this->lTiempoActual->Caption =  "Tiempo Actual :" + IntToStr(this->alg-
>tActual);
  else
    this->lTiempoActual->Caption =  "Tiempo Actual :?";
}
//---------------------------------------------------------------------------
void __fastcall TMain::Reiniciar1Click(TObject *Sender)
{
  if (this->alg == NULL){
    ShowMessage("El algoritmo no esta inicializado");
    return;
  }
  this->alg->reset();
  this->updateTiempoActual();
  this->updateTablaTareas();
  this->OpenGLAPPanel1Paint(Sender);
}
//---------------------------------------------------------------------------
void __fastcall TMain::Ejecutar1Click(TObject *Sender)

{
  if (this->alg == NULL) {
    ShowMessage("El algoritmo no esta inicializado");
    return;
  }
  if (this->alg->run()<0)
    ShowMessage("La ejecucion del algoritmo ha producido error");
  else
```

```
    ShowMessage("La ejecucion del algoritmo se ha realizado con exito");
  this->updateTiempoActual();
  this->updateTablaTareas();
  return;
}
//---------------------------------------------------------------------------
void __fastcall TMain::Siguienteevento1Click(TObject *Sender)
{
  if (this->alg == NULL) {
    ShowMessage("El algoritmo no esta inicializado");
    return;
  }
  int retVal = this->alg->next_event();
  if (retVal<0){
    ShowMessage("La ejecucion hasta el siguiente evento del algoritmo ha
producido error");
    return;
  }
  if (retVal == 2){
   ShowMessage("La ejecucion del algoritmo ha finalizado.");
   return;
  }
  this->updateTiempoActual();
  this->updateTablaTareas();
  this->OpenGLAPPanel1Paint(Sender);
}
//---------------------------------------------------------------------------
int __fastcall TMain::getEstado(Task t)
{
  if (this->alg == NULL) return ERR_ALG;
  if (this->alg->tActual <= t.getTStart()) return TAREA_ESPERA;
  /* Comprobamos que la tarea no este en ejecuccion, mirando si el tiempo de
  comienzo coincide*/
  list<SituacionTarea>::iterator it;
  for (it=this->alg->listaActual.begin();it!=this->alg-
>listaActual.end();it++){
    Task aux = (*it).tarea;
    if ((aux.getLength() == t.getLength()) &&
          (aux.getH() == t.getH()) &&
          (aux.getW() == t.getW())
        )
     return TAREA_EJEC;
  }
  return TAREA_LIB;
}
//---------------------------------------------------------------------------
void __fastcall TMain::NuevaFPGA1Click(TObject *Sender)
{
  AnsiString datos;
  if (FormFPGA->ShowModal() == mrOk){
    int H,W;
    datos = FormFPGA->tNombreFPGA->Text;
    string nombre = datos.c_str();
    datos = FormFPGA->tAltoFPGA->Text;
    try {
      H = StrToInt(datos);
    }
    catch (Exception &error){
      ShowMessage("El alto de la FPGA tiene que ser un numero entero.");
    }
    datos = FormFPGA->tAnchoFPGA->Text;
    try {
```

```
      W = StrToInt(datos);
    }
    catch (Exception &error){
      ShowMessage("El ancho de la FPGA tiene que ser un numero entero.");
    }
    if (this->fpga != NULL) delete this->fpga;
    this->fpga = new FPGA(W,H,nombre);
    // si la lista de tareas no esta vacia hay que reiniciar el algoritmo
    if (this->listaTareas.size() != 0) {
      this->alg = new BinPackingMER();
      this->alg->Init(this->listaTareas,this->fpga->getH(),this->fpga->getW(),
                      this->fpga->getName());
      this->updateTiempoActual();
    }
    this->updateDatosFPGA();
    // Cambiamos la Fpga en el panel
    this->setPanelFPGA();

  }

}
//---------------------------------------------------------------------------
void __fastcall TMain::updateDatosFPGA()
{
  this->Label1->Caption = "Datos FPGA:";
  if (this->fpga == NULL){
    this->lAnchoFPGA->Caption = "Ancho:?" ;
    this->lAltoFPGA->Caption = "Alto:?" ;
    this->lNombreFPGA->Caption = "Nombre:?";
  }
  else{
    this->lAnchoFPGA->Caption = "Ancho:" + IntToStr(this->fpga->getW());
    this->lAltoFPGA->Caption = "Alto:" + IntToStr(this->fpga->getH());
    this->lNombreFPGA->Caption = "Nombre:" + AnsiString(this->fpga-
>getName().c_str());
  }
}
//---------------------------------------------------------------------------
void __fastcall TMain::Verhistrico1Click(TObject *Sender)
{
  ifstream is("Historico.txt");
  if (!is){
    ShowMessage("Error al abrir el fichero de historia");
    return;
  }
  char c;
  AnsiString aux;
  while (is.get(c)){
    aux += AnsiString(c);
  }
  is.close();
  FormHistorico->MemoHistorico->Text = aux;
  FormHistorico->ShowModal();
}
//---------------------------------------------------------------------------
void __fastcall TMain::GLScene()
{
  // Aqui dibujaremos la escena de marras
  // Si la Fpga no esta inicializada o esta vacia
  if ((this->fpga == NULL) ||
    ((this->fpga->getH() == 0) && (this->fpga->getW() == 0)))
    return;
```

```
  this->dibujarFPGA();
  this->dibujarTareas();
}
//---------------------------------------------------------------------------
void __fastcall TMain::OpenGLAPPanel1Paint(TObject *Sender)
{
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   GLScene();
   glFlush();
   this->OpenGLAPPanel1->SwapBuffers();

}
//---------------------------------------------------------------------------

void __fastcall TMain::OpenGLAPPanel1Init(TObject *Sender)
{
  glShadeModel(GL_SMOOTH);
      glClearColor(0.0f, 0.0f, 0.0f, 0.5f);
      glClearDepth(1.0f);
      glEnable(GL_DEPTH_TEST);
      glDepthFunc(GL_LEQUAL);
      glHint(GL_PERSPECTIVE_CORRECTION_HINT, GL_NICEST);
}
//---------------------------------------------------------------------------
void __fastcall TMain::OpenGLAPPanel1Resize(TObject *Sender)
{
   GLfloat nRange = 200.0;
   w = this->OpenGLAPPanel1->Width;
   h = this->OpenGLAPPanel1->Height;

   if (h == 0) h = 1; // Prevenimos la division por cero
   if (w == 0) w = 1;

   glViewport(0,0,w,h);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   if (w<=h)
   {
      this->left = -nRange;
      this->rigth = nRange;
      this->top = nRange * h/w;
      this->bottom = -nRange * h/w;
   }
   else
   {
      this->left = -nRange*w/h;
      this->rigth = nRange*w/h;
      this->top = nRange;
      this->bottom = -nRange;
   }
   gluOrtho2D(this->left,this->rigth,this->bottom,this->top);
   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
}
//---------------------------------------------------------------------------
void __fastcall TMain::dibujarFPGA(){
  glColor3f(1.0f,1.0f,1.0f);
  glBegin(GL_LINES);
  for (int i=0;i<this->fpga->getW();i++){
    glVertex2i(i,0);
    glVertex2i(i,this->fpga->getH());
  }
```

```
  glEnd();
  glBegin(GL_LINES);
  for (int i=0;i<this->fpga->getH();i++){
    glVertex2i(0,i);
    glVertex2i(this->fpga->getW(),i);
  }
  glEnd();
}
//---------------------------------------------------------------------------
void __fastcall TMain::dibujarTareas(){
  // Si no esta inicado el algoritmo,nos salimos
  if (this->alg == NULL) return;
  list<SituacionTarea>::iterator it;
  for (it=this->alg->listaActual.begin();it != this->alg-
>listaActual.end();it++){
    SituacionTarea st = (*it);
    Task t = st.tarea;
    glBegin(GL_QUADS);
      glColor3ub(st.r,st.g,st.b);
      glVertex2i(st.x,st.y);
      glVertex2i(st.x + t.getW(),st.y);
      glVertex2i(st.x + t.getW(),st.y + t.getH());
      glVertex2i(st.x,st.y + t.getH());
    glEnd();
  }
}
//---------------------------------------------------------------------------
// Cambia los ejes de coordenadas del panel, para adaptarlo a la nueva FPGA
void __fastcall TMain::setPanelFPGA()
{
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  gluOrtho2D(0,this->fpga->getW(),0,this->fpga->getH());
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
}
//---------------------------------------------------------------------------
bool __fastcall TMain::checkDatosFPGA(FPGA fpga,int minH,int maxH,int minW,
                                      int maxW,int numTareas,int tiempoFinal)
{
  if (minH <= 0){
    ShowMessage("¡Error!. La altura minima debe ser mayor que cero.");
    return false;
  }
  if (minW <= 0){
    ShowMessage("¡Error!. La anchura minima debe ser mayor que cero.");
    return false;
  }
  if (maxH > fpga.getH()){
    ShowMessage("¡Error!. La altura maxima es mayor que la de la fpga.");
    return false;
  }
  if (maxW > fpga.getW()){
    ShowMessage("¡Error!. La anchura maxima es mayor que la de la fpga.");
    return false;
  }
  if (numTareas<=0){
    ShowMessage("¡Error!. El numero de tareas debe ser mayor que cero");
    return false;
  }
  if (tiempoFinal<numTareas){
```

```
    ShowMessage("¡Error!. El tiempo final debe ser mayor o igual que el numero
de tareas");
    return false;
  }
  return true;
}
//---------------------------------------------------------------------------
void __fastcall TMain::FormDestroy(TObject *Sender)
{
  if (this->fpga != NULL) delete this->fpga;
  if (this->log != NULL) delete this->log;
  if (this->alg != NULL) delete this->alg;
}
//---------------------------------------------------------------------------
// Metodo para actualizar la barra de estado de la ventana
void __fastcall TMain::updateStatusBar(){
  AnsiString cadena;
  if (this->alg == NULL)
    cadena = "Algoritmo no seleccionado.";
  else {
    cadena = AnsiString(this->alg->comentario.c_str());
    cadena += "Ocupacion Instantanea: " + AnsiString(this->alg-
>getOcupacionInst()) + " % ";
    cadena += "Ocupacion Media: " + AnsiString(this->alg->getOcupacionMedia())
+ " %";
  }
  this->barraEstado->SimpleText = cadena;
}


void __fastcall TMain::AbrirConfiguracion1Click(TObject *Sender)
{
string nomAlg;
bool algOK=false;
  if (AbrirConf->Execute()) {
    if (this->fpga == NULL) this->fpga = new FPGA();
    if (loadConf(AbrirConf->FileName.c_str(),this->listaTareas,*fpga,nomAlg)
<0){
      ShowMessage("Error al cargar la configuración ");
    return;
    }
    else {
      ShowMessage("¡Configuración cargada con exito!");
      this->updateDatosFPGA();
      this->setPanelFPGA();
      this->updateTablaTareas();
      this->updateTablaTareas();
      if (nomAlg == "BinPacking MER "){
          this->alg = new BinPackingMER();
          algOK=true;
      }
// ****************************************
// Aquí cargaríamos el resto de los algoritmos
// ****************************************

      if (algOK){
        this->alg->Init(this->listaTareas,this->fpga->getH(),this->fpga-
>getW(),
                    this->fpga->getName());
        this->updateTiempoActual();
       this->updateStatusBar();
      }
```

```
      else {
        ShowMessage("Fallo al cargar el Algoritmo: El nombre no corresponde a
ningún algoritmo del sistema. Por favor cargue uno manualmente");
      }
    }

  }

}
//---------------------------------------------------------------------------

void __fastcall TMain::GuardarConfiguracion1Click(TObject *Sender)
{
bool hayAlg;
bool hayFPGA=false;
bool hayLista=false;
try {
    if(this->alg->comentario.length()>0){hayAlg=true;}
    }
catch (Exception &error){
    hayAlg=false;
}
if (this->fpga->getH()>0 && this->fpga->getW()>0){hayFPGA=true;}
if (this->listaTareas.size() >0){hayLista=true;}
if (!hayAlg) ShowMessage("Tienes que seleccionar un Algoritmo para poder
guardar la configuración");
if (!hayFPGA) ShowMessage("Tienes que seleccionar una FPGA para poder guardar
la configuración");
if (!hayLista) ShowMessage("Tienes que seleccionar una Lista de Tareas para
poder guardar la configuración");
if (hayFPGA && hayLista && hayAlg){
    if (GuardarConf->Execute()) {
      if (saveConf(GuardarConf->FileName.c_str(),this->listaTareas,*fpga,this-
>alg) <0){
          ShowMessage("Error al guardar la configuración");
          return;
      }
    else ShowMessage("¡Configuración guardada con exito!");
    }
  }
}
//---------------------------------------------------------------------------
```

Fichero Project1.cpp

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop
USEFORM("DatosLista.cpp", ListaTareas);
USEFORM("Acerca.cpp", Acercade);
USEUNIT("Utils.cpp");
USEUNIT("BinPackingMER.cpp");
USEUNIT("Log.cpp");
USEUNIT("CRectangle.cpp");
USEUNIT("Task.cpp");
USEUNIT("StreamTokenizer.cpp");
USEUNIT("RGBColor.cpp");
USEUNIT("FPGA.cpp");
USEFORM("DatosFPGA.cpp", FormFPGA);
USEFORM("VerHistorico.cpp", FormHistorico);
```

```
USEFORM("Ppal.cpp", Main);
USERES("Project1.res");
//---------------------------------------------------------------------------
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
  try{
    Application->Initialize();
    Application->Title = "FPGA Simulator";
     Application->HelpFile = "C:\\Documents and Settings\\chema\\Mis
documentos\\My Projects\\c++\\binpacking2d\\src\\Manual de usuario.chm";
     Application->CreateForm(__classid(TMain), &Main);
     Application->CreateForm(__classid(TListaTareas), &ListaTareas);
     Application->CreateForm(__classid(TAcercade), &Acercade);
     Application->CreateForm(__classid(TFormFPGA), &FormFPGA);
     Application->CreateForm(__classid(TFormHistorico), &FormHistorico);
     Application->Run();
  }
  catch (Exception &exception){
    Application->ShowException(&exception);
  }
  return 0;
}
//---------------------------------------------------------------------------
```

Fichero RGBColor.cpp

```
#include "RGBColor.h"
RGBColor::RGBColor()
{
        r = g = b = 0;
}
//---------------------------------------------------------------------------
RGBColor::RGBColor(RGBColor &color)
{
        r = color.r;
        g = color.g;
        b = color.b;
}
//---------------------------------------------------------------------------
```

Fichero StreamTokenizer.cpp

```
#include "StreamTokenizer.h"
using namespace std;
string StreamTokenizer::next() {
      string result;
      if(p != end) {
            insert_iterator<string> ii(result, result.begin());
            while(isDelimiter(*p) && p != end) p++;
            while (!isDelimiter(*p) && p != end) *ii++ = *p++;
      }
      return result;
}
```

Fichero Task.cpp

```
#include "Task.h"
```

```
#include <strstream>
using namespace std;
//-----------------------------------------------------------------
Task::Task(int w,int h,int tStart,int length,int tMax){
      this->w = w;
      this->h = h;
      this->tStart = tStart;
      this->length = length;
      this->tMax = tMax;
}
//-----------------------------------------------------------------
Task::Task(){
      this->w = 0;
      this->h = 0;
      this->tStart = 0;
      this->length = 0;
      this->tMax = 0;
}
//-----------------------------------------------------------------
Task::Task(const Task& t){ // Constructor de copia
      this->w = t.w;
      this->h = t.h;
      this->tStart = t.tStart;
      this->length = t.length;
      this->tMax = t.tMax;
}
//-----------------------------------------------------------------
Task::~Task(){
//    delete color;
}
//-----------------------------------------------------------------
int Task::getW(){ return this->w; }
//-----------------------------------------------------------------
int Task::getH() { return this->h; }
//-----------------------------------------------------------------
int Task::getTStart() { return this->tStart; }
//-----------------------------------------------------------------
int Task::getLength() {return this->length; }
//-----------------------------------------------------------------
int Task::getTMax(){ return this->tMax; } // FIX:Por ahora no se utiliza
//-----------------------------------------------------------------
void Task::setColor(RGBColor color){ this->color = new RGBColor(color); }
//-----------------------------------------------------------------
void Task::getColor(RGBColor& color) {color = *(this->color) ; }
//-----------------------------------------------------------------
string Task::toString(){
      ostrstream s;
      s << "Clase Task:" << endl;
      s << "Ancho = " << this->getW() << endl;
      s << "Alto = " << this->getH() << endl;
      s << "TStart = " << this->getTStart() << endl;
      s << "Length = " << this->getLength() << endl;
      s << "TMax = " << this->getTMax() << endl;
      s << ends;
      string retVal;
      retVal = s.str();
      // Cosas que hay que hacer para liberar el stream
      s.seekp(-1, ios::cur);
      s.rdbuf()->freeze(0);
      return retVal;
}
//-----------------------------------------------------------------
```

Fichero Utils.cpp

```
#include "Utils.h"
#include "Task.h"
#include "StreamTokenizer.h"
#include <vcl.h>
#include <math.h>
#include <string>
#include <set>
#include <iostream>
#include <fstream>
#include <cstdlib> //Para generar numeros aleatorios
#include <ctime> // Para generar la semilla aleatoria
using namespace std;
//---------------------------------------------------------------------------
---------------
/* Crea una lista de tareas aleatorias
* Valor de retorno: 0, en caso de exito, -1 en caso de error
*/
int createTasks(int minW,int maxW,int minH,int maxH,int maxTimeTask,int
maxTime,
                int numTask,FPGA fpga,list<Task>& lTask){
      // if (DEBUG) ECHO("Entrando al metodo createTask");
      lTask.clear(); // limpiamos lo que hubiera en la lista de entrada
      // if (DEBUG) ECHO("Creando las tareas");
      // Comprobamos los parametros:
      if (minW <= 0) {
            // if (DEBUG)WARNING("Parametro minW menor que cero. Se pone a
uno");
            minW = 1;
      }
      if (minH<= 0) {
            // if (DEBUG)WARNING("Parametro minH menor que cero. Se pone a
uno");
            minH = 1;
      }
      if (maxW > fpga.getW()) {
            // if (DEBUG)WARNING("Parametro maxW mayor que el tamaño de la
fpga.Se pone al tamaño de la fpga");
            maxW = fpga.getW();
      }
      if (maxH > fpga.getH()) {
            // if (DEBUG)WARNING("Parametro maxH mayor que el tamaño de la
fpga.Se pone al tamaño de la fpga");
            maxH = fpga.getH();
      }
      if ((minW >= maxW) || (minH >= maxH)){
            // if (DEBUG)WARNING("Algun parametro minimo es mayor o igual que
el maximo. Salimos de la funcion");
            return -1;
      }
      if (numTask <= 0) {
            // if (DEBUG)WARNING("Numero  de tareas menor o igual que cero.");
            return -1;
      }
  if (maxTime < numTask) {
      if (DEBUG) WARNING("Tiempo final menor que el numerode tareas");
            return -1;
  }
      // if (DEBUG) ECHO("Parece que todos los parametros son correctos");
```

```
      // Diferencias entre los minimos y los maximos de ancura y altura
      int difW = maxW - minW + 1;
      int difH = maxH - minH + 1;
      // Sembramos la semilla aleatoria
      srand(time(0));
      for (int i=0;i< numTask;i++){
            //if (DEBUG) ECHO("Generando anchura");
            int W = rand() % difW;
            W += minW; // numero entre minW maxW
            //if (DEBUG) ECHO("Generando altura");
            int H = rand() % difH;
            H += minH; // numero entre minH y maxH
            int length = (rand() % maxTimeTask)  + 1;
            int time;
    bool esta;
    // Codigo de prueba para que no lleguen dos tareas en el mismo tiempo
    do {
      esta = false;
      time = rand() % maxTime; // 0 -- maxTime
      list<Task>::iterator it  = lTask.begin();
      while ((!esta) && (it != lTask.end())) {
        if ((*it).getTStart() == time) esta = true;
        it++;
      }
    } while (esta);
            //if (DEBUG) ECHO("insertado el tiempo en el conjunto");
            Task t(W,H,time,length,0); // TODO: Falta el ultimo parametro que
se deja por defecto a cero (TMax)
            lTask.push_back(t); // Insertamos en la lista
            //if (DEBUG) ECHO("insertado el tiempo en la lista");
      }
      // if (DEBUG) ECHO("Salimos del bucle");
      lTask.sort(); // Ordenamos la lista de tareas por tiempo de llegada
      return 0;
}
//-------------------------------------------------------------------------
---------------
/* Carga una lista de tareas de un fichero.
* TODO : Las lineas que comienzen con # seran consideradas
* un comentario, por lo que se ignoraran. Para hacerlo hace falta leer el
fichero de linea en linea
* Valor de retorno: 0, en caso de exito, -1 en caso de error
*/
int loadTasks(char *path,list<Task>& lTask){
      lTask.clear(); // limpiamos las tareas que hubiera en la lista
      ifstream in(path);
      if (!in){
            WARNING_VAR("No se pudo abrir el fichero " , path);
            return -1;
      }
      StreamTokenizer words(in); // Para parsear el fichero palabra por
palabara
      string word; // cada palabra del fichero
      int i =0;
      int H,W,tStart,length,tMax; // Variables auxiliares
      while((word = words.next()).size() != 0) {
            //if (DEBUG) ECHO_VAR("Palabra extraida = ",word);
            switch (i){
                  case(0):
                        W = atoi(word.c_str());
                        i++;
                        break;
```

```
                  case(1):
                        H = atoi(word.c_str());

                        i++;
                        break;
                  case(2):
                        tStart = atoi(word.c_str());
                        i++;
                        break;
                  case(3):
                        length = atoi(word.c_str());
                        i++;
                        break;
                  case(4):
                        tMax = atoi(word.c_str());
                        i++;
                        break;
                  default:
                        return -1;

            }
            if (i==5){
                  Task t(W,H,tStart,length,tMax);
                  lTask.push_back(t);
                  i = 0;
            }
      }
      in.close();
      return 0;
}
//-------------------------------------------------------------------------
---------------
/* Salva a disco una lista de tareas.
* Valor de retorno: 0, en caso de exito, -1 en caso de error
*/
int saveTasks(char *path,list<Task>& lTask){
      if (lTask.size() == 0) {
            WARNING("La lista de tareas esta vacia.");
            return -1;
      }
      ofstream of(path); // Output Stream para escribir al fichero
      if (!of){
            WARNING_VAR("No se pudo crear el fichero en la ruta ",path);
            return -1;
      }
      for(list<Task>::iterator it=lTask.begin();it != lTask.end();it++){
            Task t = *it; // Conseguimos la tarea actual
            of << t.getW() << " "  << t.getH() << " " << t.getTStart() << " "
<< t.getLength() << " "<< t.getTMax() << endl;
      }
      of.flush();
      of.close();
      return 0;
}
//-----------------------------------------------------------------
/** Metodo para hallar la interseccion entre dos rectas dadas por dos puntos
Devuelve -1 si las rectas son paralelas, es decir si el punto de interseccion
no es valido
*/
int interseccion(Point2D a,Point2D b,Point2D c,Point2D d,Point2D &i){
      // if (DEBUG) ECHO("Entrando al metodo interseccion de Utils");
      // Resolvemos la ecuacion de las rectas
```

```
      // Ponemos cada recta de la forma ax + by + c = 0
      Point2D v1 = b - a; // vector director de la recta 1
      Point2D v2 = d - c; // vector director de la recta 2
      int A,B,C,D,E,F; // Coeficientes de las ecuaciones de las rectas
      A = v1.getY();
      B = -v1.getX();
      C = (a.getY() * v1.getX()) - (a.getX() * v1.getY());
      D = v2.getY();
      E = -v2.getX();
      F = (c.getY() * v2.getX()) - (c.getX() * v2.getY());
      /*
      if (DEBUG) {
            cout << "Valor de las rectas A="<< A << ",B=" << B << ",C=" << C
<< endl;
            cout << "Valor de las rectas D="<< D << ",E=" << E << ",F=" << F
<< endl;
      }
      */
      // Miramos si las rectas son paralelas
      if (paralelas(v1,v2)) {
            // if (DEBUG) ECHO("Las rectas son PARALELAS");
            return -1;

      }
      if (coincidentes(A,B,C,D,E,F)){
            // if (DEBUG) ECHO("Las rectas son COINCIDENTES");
            return -1;
      }
      float x,y; // Coordenadas del punto de interseccion
      if (A == 0.0f) {// La primera recta es horizontal
            y = -C / B;
            x = (-F - E*y) / D;
      }
      else
      if (D == 0.0f) {// La segunda recta es horizontal
            y = -F / E;
            x = (-C - B*y) / A;
      }
      else
      if (B == 0.0f){ // la primera recta es vertical
            x = -C / A;
            y = (-F -D*x) / E;
      }
      else
      if (E == 0.0f){ // La segunda recta es vertical
            x = -F / D;
            y = (-C -A*x) / B;
      }
      else{
            float aux1 = F*B - C*E;
            float aux2 = A*E - D*B;
            x = aux1 / aux2;
            y = (-C - A*x) / B;
      }
      /*
      if (DEBUG){
            ECHO_VAR("La coordenada x del punto es =",x);
            ECHO_VAR("La coordenada y del punto es =",y);
      }
      */
      i.setX(x);
      i.setY(y);
```

```
	return 0;
}
//-------------------------------------------------------------
/** Parametros :
Damos las rectas de la forma :
Dos rectas son paralelas sii sus vectores de direccion forman 180° */
bool paralelas(Point2D v1,Point2D v2){
	float m1 = sqrt(pow(v1.getX(),2.0f) + pow(v1.getY(),2.0f)); // Modulo
del vector 1
	float m2 = sqrt(pow(v2.getX(),2.0f) + pow(v2.getY(),2.0f)); // Modulo
del vector 2
	// if (DEBUG) ECHO_VAR("Modulo 1=",m1);
	// if (DEBUG) ECHO_VAR("Modulo 2=",m2);
	float prodEsc = v1.getX() * v2.getX() + v1.getY() * v2.getY();
	float prodMod = m1 * m2;
	/*
	if (DEBUG) {
		ECHO_VAR("producto escalar=",prodEsc);
		ECHO_VAR("Producto de modulos=",prodMod);
	}
	*/
	if (prodEsc == prodMod) return true;
	else return false;
}
//-------------------------------------------------------------
/** Parametros :
Damos las rectas de la forma :
Ax + By + C = 0
Dx + Ey + F = 0
Dos rectas son coindentes sii A/D = B/E == C/F */
bool coincidentes(int A,int B,int C,int D,int E,int F){
	float aux1 = (D == 0) ? (float) A : (float) A / D;
	float aux2 = (E == 0) ? (float) B: (float) B/ E;
	float aux3 = (F == 0) ? (float) C: (float) C/ F;
	/*
	if (DEBUG) {
		ECHO("Metodo coincidentes");
		ECHO_VAR("Valor de aux1=",aux1);
		ECHO_VAR("Valor de aux2=",aux2);
		ECHO_VAR("Valor de aux3=",aux3);
	}
	*/
	if ((aux1 == aux2) && (aux2 == aux3)) return true;
	else return false;
}
//-------------------------------------------------------------
void intercambiar(int &a,int &b){
	int aux = a;
	a = b;
	b = aux;
	return;
}
//-----------------------------------------------------------
// Guarda una fpga en un fichero
int saveFPGA(char *path,FPGA fpga){
	ofstream of(path); // Output Stream para escribir al fichero
	if (!of){
		WARNING_VAR("No se pudo crear el fichero en la ruta ",path);
		return -1;
	}
	of << fpga.getH() << " "  << fpga.getW() << " " <<
fpga.getName().c_str() << endl;
```

```
      of.flush();
      of.close();
      return 0;
}
//-------------------------------------------------------------------------
-
// Carga una fpga desde disco
int loadFPGA(char *path,FPGA &fpga){
  ifstream in(path);
      if (!in){
            WARNING_VAR("No se pudo abrir el fichero " , path);
            return -1;
      }
      StreamTokenizer words(in); // Para parsear el fichero palabra por
palabra
      string word; // cada palabra del fichero
      int i =0;
      int H,W; // Variables auxiliares
  string name;
      while((word = words.next()).size() != 0) {
            //if (DEBUG) ECHO_VAR("Palabra extraida = ",word);
            switch (i){
                  case(0):
                        H = atoi(word.c_str());
                        i++;
                        break;
                  case(1):
                        W = atoi(word.c_str());
                        i++;
                        break;
                  case(2):
                        name = word;
                        i++;
                        break;
            }
            if (i==3){
      fpga.setH(H);
      fpga.setW(W);
      fpga.setName(name);
      i = 0;
            }
      }
      in.close();
      return 0;
}
//-------------------------------------------------------------------------
-
int loadConf(char *path,list<Task>& lTask,FPGA &fpga,string &alg){
 ifstream in(path);
      if (!in){
            WARNING_VAR("No se pudo abrir el fichero " , path);
            return -1;
      }
      StreamTokenizer words(in); // Para parsear el fichero palabra por
palabra
      string word; // cada palabra del fichero
      int i =0;
      //int H,W; // Variables auxiliares
  int H,W,tStart,length,tMax;
  string name;
  AnsiString cadena;
  cadena=word.c_str();
```

```
  word = words.next();
  cadena=word.c_str();
  //ShowMessage(cadena);

 if  (word=="FPGA"){
ShowMessage("FPGA cargada con éxito...");
      while((word = words.next()) != "TAREAS") {
            switch (i){
                  case(0):
                        H = atoi(word.c_str());
                        i++;
                        break;
                  case(1):
                        W = atoi(word.c_str());
                        i++;
                        break;
                  case(2):
                        name = word;
                        i++;
                        break;
            }
            if (i==3){
      fpga.setH(H);
      fpga.setW(W);
      fpga.setName(name);
      i = 0;
            }
  }
 }
 if  (word=="TAREAS"){
 i=0;
 lTask.clear();
  ShowMessage("Tareas cargadas con éxito...");
    while((word = words.next()) != "ALGORITMO") {
            switch (i){
                  case(0):
                        W = atoi(word.c_str());
                        i++;
                        break;
                  case(1):
                        H = atoi(word.c_str());
                        i++;
                        break;
                  case(2):
                        tStart = atoi(word.c_str());
                        i++;
                        break;
                  case(3):
                        length = atoi(word.c_str());
                        i++;

                        break;
                  case(4):
                        tMax = atoi(word.c_str());
                        i++;
                        break;
                  default:
                        return -1;
            }
            if (i==5){
                  Task t(W,H,tStart,length,tMax);
                  lTask.push_back(t);
```

```
                  i = 0;
            }
      }
 }
 if (word=="ALGORITMO"){
ShowMessage("Algoritmo cargado con éxito...");
  ostrstream s;
  while((word = words.next()) != "END") {
      s << word << " ";
  }
      s << ends;
      alg = s.str();
      s.seekp(-1, ios::cur);
      s.rdbuf()->freeze(0);
 }
 ShowMessage(alg.c_str());
 in.close();

      return 0;
}

//-------------------------------------------------------------------------------
-
int saveConf(char *path,list<Task>& lTask,FPGA fpga,Algoritmo *alg){
      ofstream of(path); // Output Stream para escribir al fichero
      if (!of){
            WARNING_VAR("No se pudo crear el fichero en la ruta ",path);
            return -1;
      }
  AnsiString cadena;
  cadena = AnsiString(alg->comentario.c_str());
  of << "FPGA" << endl;
      of << fpga.getH() << " "  << fpga.getW() << " " <<
fpga.getName().c_str() << endl;
  of << "TAREAS" << endl;
      for(list<Task>::iterator it=lTask.begin();it != lTask.end();it++){
            Task t = *it; // Conseguimos la tarea actual
            of << t.getW() << " "  << t.getH() << " " << t.getTStart() << " "
<< t.getLength() << " "<< t.getTMax() << endl;
      }
  of << "ALGORITMO" << endl;
  of << cadena << endl;
  of << "END" << endl;
      of.flush();
      of.close();
return 0;
}
//-------------------------------------------------------------------------------
-
float round2Dec(float valor){
    float retVal = (float) 0;
    int parteEntera = (int) valor;
    float parteDecimal = valor - parteEntera;
    parteDecimal *= 100;
    int aux = (int) parteDecimal;
    float aux2 =  ((float)aux / 100.0f);
    retVal = (float)parteEntera + aux2;
    return retVal;
}
```

Fichero VerHistorico.cpp

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "VerHistorico.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TFormHistorico *FormHistorico;
//---------------------------------------------------------------------------
__fastcall TFormHistorico::TFormHistorico(TComponent* Owner)
  : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TFormHistorico::AceptarClick(TObject *Sender)
{
  this->ModalResult = mrOk;
}
//---------------------------------------------------------------------------
```

ANEXO 3

MANUAL DE USUARIO

Manual de usuario
Tabla de contenidos

¿Qué es FPGA Simulator?

FPGA Simulator 1.0 es un programa que nos permite simular el comportamiento de una FPGA bidimensional en un entorno controlado.

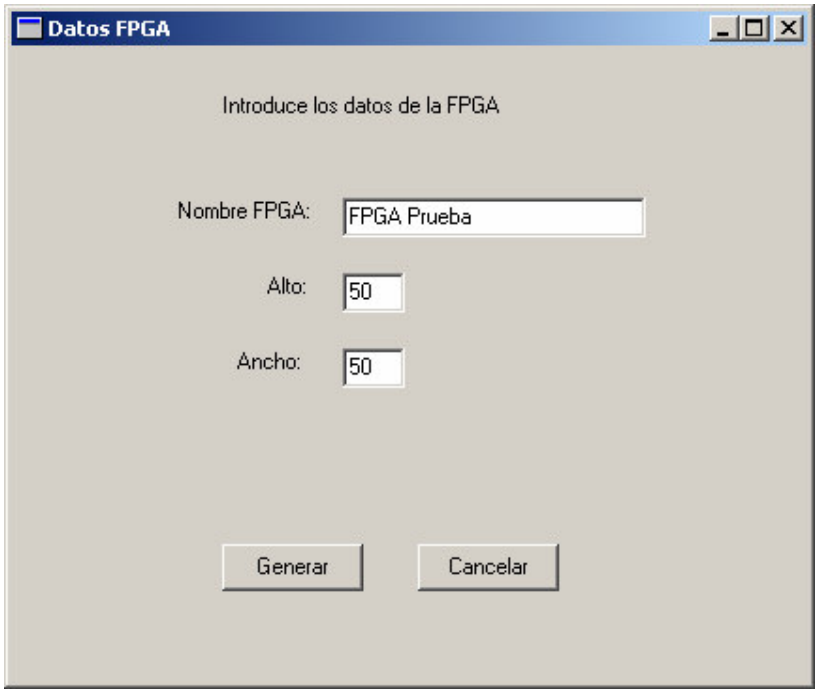
Parámetros configurables

FPGA Simulator 1.0 nos permite configurar distintos aspectos de la simulación. Los parámetros configurables son los siguientes:

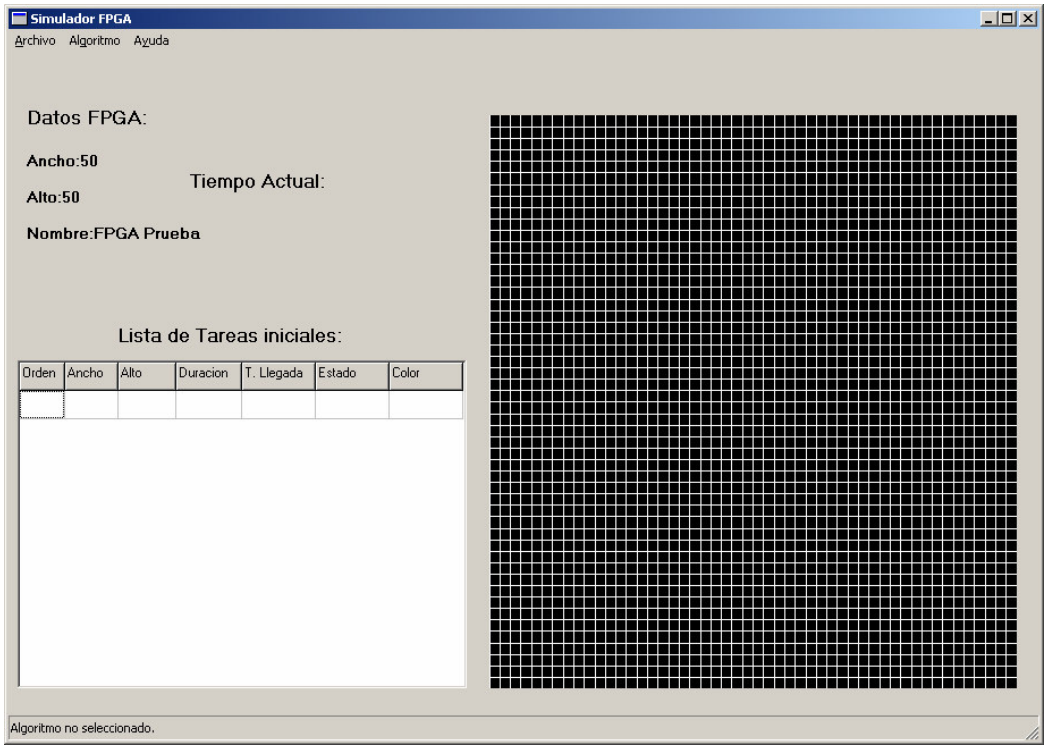
- FPGA.
- Lista de Tareas
- Algoritmo

Configurar una FPGA

Para poder trabajar con el programa, necesitamos primero crear una FPGA sobre la cual simular todo el proceso. Para hacer esto pinchamos en **Archivo → Nueva FPGA**. Aparecerá la siguiente pantalla:



Los datos necesarios para crear una FPGA son el nombre, el alto y el ancho. En este caso hemos llamado a la FPGA *FPGA Prueba* y su tamaño es de 50x50 unidades.
Pinchamos en el botón Generar y la FPGA ya estará creada.
Véase que en la pantalla principal aparecen los parámetros de nuestra nueva FPGA.



Guardar y abrir archivos de FPGA

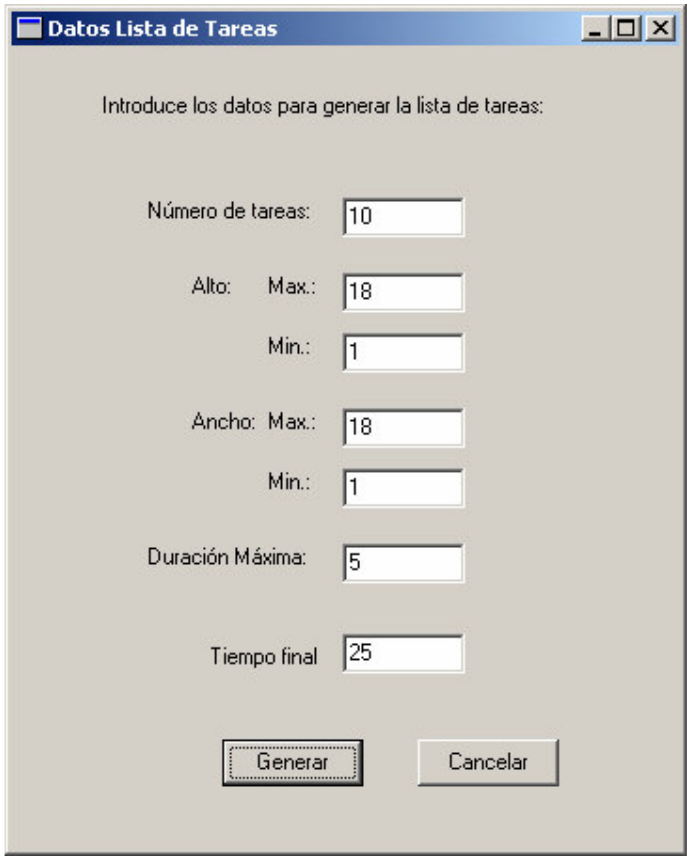
Las configuraciones de FPGA se pueden guardar en un archivo de manera que puedan ser recuperados posteriormente. Para guardar, seleccionar **Archivo → Guardar FPGA**. Se abrirá un diálogo que nos pedirá el nombre y la ubicación del fichero de configuración de la FPGA.
Por defecto, los archivos de configuración de FPGA llevan la extensión *.fpg*.
Para recuperar un archivo creado con anterioridad, acceder al menú **Archivo → Abrir FPGA**. Una ventana de diálogo nos pedirá el archivo. Pinchar en aceptar y la FPGA quedará cargada en el programa.

Configurar una lista de tareas

Para poder simular un proceso en una FPGA, necesitamos tareas que se vayan ubicando en la FPGA. Las listas de tareas cumplen una serie de requisitos:

- Cuando una tarea se ubica en la FPGA, ésta se ejecuta *ocupando* un espacio rectangular.
- Cuando una tarea acaba de ejecutarse, sale de la FPGA dejando libre el espacio que ocupaba, de manera que otra tarea nueva pueda ocupar su sitio.
- Una tarea no puede ocupar un espacio en la FPGA ocupado por otra tarea.
- Las tareas no pueden ocupar un espacio exterior a la FPGA.
- La ubicación de las tareas en la FPGA está determinado por un algoritmo interno al programa configurable.
- Cuando una tarea no tiene sitio en la FPGA para ejecutarse, espera hasta que sea posible su ubicación.

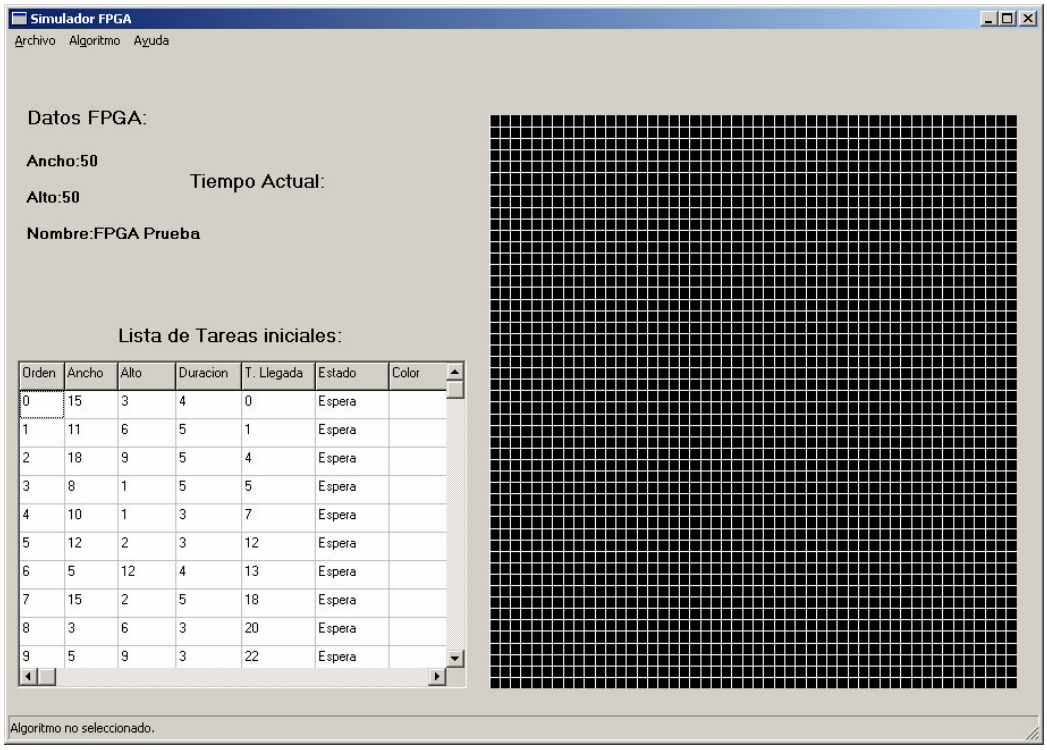
Para crear una nueva lista de tareas aleatorias, debemos introducir una serie de rangos entre los cuales oscilan las tareas. Pinchar en **Archivo → Nueva Lista de Tareas**. Aparecerá una ventana de diálogo que nos pedirá todos los datos necesarios:



Los datos requeridos son los siguientes:

- Número de tareas: Número de tareas que contendrá la lista.
- Alto máximo y mínimo: Valores máximo y mínimo entre los que oscilarán las alturas de las tareas, escogidas al azar.
- Ancho máximo y mínimo: Igual que el anterior, pero respecto al ancho de las tareas.
- Duración máxima: La duración de las tareas en la FPGA es un valor aleatorio comprendido entre 0 y la duración máxima.
- Tiempo final: El tiempo de llegada de las tareas oscila entre 0 y este valor. En un mismo momento sólo puede llegar una tarea a la FPGA.

Una vez introducidos todos los datos, sólo nos queda generar la lista. Vemos que los cambios también se reflejan en la pantalla principal, en la rejilla de Lista de tareas iniciales :



Guardar y abrir listas de tareas

Al igual que las FPGAs, las listas de tareas también se pueden guardar en un archivo, que por defecto tendrá la extensión *.tsk*.
El procedimiento para guardar una lista de tareas es análogo al de la FPGA. Seleccionar en el menú **Archivo → Guardar Lista de tareas**. Aparecerá una ventana que nos pedirá el nombre y la ubicación del archivo.
Para abrir archivos que contengan listas de tareas, seleccionar en el menú **Archivo → Abrir Listas de tareas**. En la ventana que aparecerá seleccionar el archivo y pulsar en Aceptar. La lista de tareas se cargará automáticamente.

Último paso para la simulación: Seleccionar el algoritmo

Para poder realizar la simulación, necesitamos seleccionar el tipo de algoritmo que queremos para ubicar las tareas dentro de la FPGA a medida que van llegando. Para hacer esto pinchar en **Algoritmo → Seleccionar...**
Por defecto, el algoritmo seleccionado es el MER (Máximum Empty Rectangle) que coloca las tareas en la esquina inferior izquierda del mayor rectángulo disponible (libre) en la FPGA.
Una vez seleccionado el algoritmo, el tiempo se pondrá a 0, esperando a que se empiece la simulación.

Configuraciones de Simulación

Las configuraciones de simulación consisten en el conjunto formado por FPGA, Lista de Tareas y Algoritmo de ubicación. Este entorno de simulación puede ser guardado en un fichero y recuperado posteriormente de manera que se puedan reproducir las simulaciones, además de convertirlos en elementos portables.

Guardar y abrir configuraciones

Para guardar una configuración seleccionar **Archivo → Guardar configuración**. El fichero que genera tiene como extensión por defecto *.cfg*. En este fichero se guardan tanto las configuraciones iniciales de la FPGA como de la lista de tareas, a la vez que el nombre del algoritmo utilizado. Para abrir una configuración de simulación, el proceso a seguir es el mismo que para el resto de elementos del programa. Seleccionar en el menú **Archivo → Abrir configuración** y escoger el fichero. Automáticamente se cargarán FPGA, tareas y algoritmo en el programa, listo para ejecución.

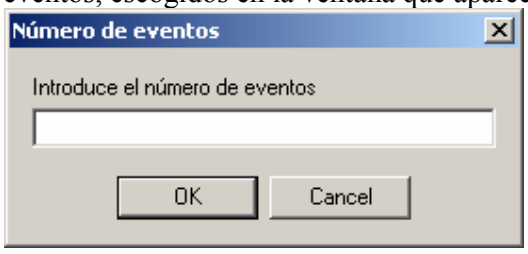
Simulación y eventos

Un elemento indispensable son los *Eventos*. Los eventos son los sucesos que se dan al ejecutar el algoritmo y que alteran el estado de la FPGA. Un evento puede estar provocado por:

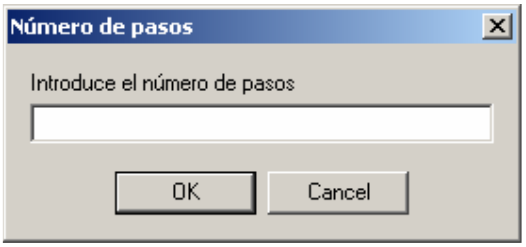
- Una tarea que entra en la FPGA y es ubicada en un cierto lugar.
- Una tarea que sale de la FPGA y que deja un espacio libre.

Existen distintos tipos de simulación en el menú **Algoritmo**:

- Simular: Simula la ejecución de las tareas en la FPGA hasta que ya no quedan tareas pendientes en la lista.
- Simular hasta: Permite distintas opciones:
  - Siguiente evento: Continúa la ejecución hasta que se produce un evento.
  - Siguiente paso: Ejecuta el algoritmo una unidad de tiempo.
  - Seleccionar eventos: Continúa la ejecución del algoritmo hasta que ocurren un número determinado de eventos, escogidos en la ventana que aparece al seleccionar esta opción:



  - Seleccionar pasos: Ejecuta el algoritmo un número determinado de unidades de tiempo que escogemos en la ventana de diálogo que aparece al escoger esta opción:



La representación de las tareas en la FPGA se hace mediante un rectángulo de un color escogido al azar: